Eötvös Loránd Tudományegyetem
Bölcsészettudományi Kar

DOKTORI DISSZERTÁCIÓ

Keszthelyi András

Adatbázis alapú optimalizáció
az oktatásszervezésben

Irodalomtudományi Doktori Iskola, vezetője:
Prof. Dr. Kulcsár Szabó Ernő

Könyvtártudomány Program, vezetője:
Prof. Dr. Sebestyén György CSc

A bizottság tagjai és tud. fokozatuk:
Dr. Komjáth Péter DSc, egyetemi tanár, elnök
Dr. Baráthné Hajdú Ágnes PhD, főiskolai tanár, bíráló
Dr. Tar József PhD, főiskolai tanár, bíráló
Dr. Kovács Valéria PhD, titkár
Dr. Szeghegyi Ágnes PhD, főiskolai tanár, tag
Dr. Fodor János PhD, póttag
Dr. Kiszl Péter PhD, póttag

Témavezető:
Prof. Dr. Sebestyén György CSc

Budapest, 2010

Alkotni születtünk erre a földre
Nemcsak megélni szép napokat
Várakat emelni a hegytetőkre
Széles folyókra nagy hidakat

Szörényi - Bródy
Kőműves Kelemen

Tartalom

1. Bevezetés

Alapprobléma

A digitálisan kezelendő adatok napjainkra mind mennyiségüket, mind változékonyságukat, mind pedig kereshetőségüket tekintve - úgy tűnik - a kezelhetetlenség határára jutottak, ugyanakkor az ezen adatoktól való függőség mértéke erősen megnövekedett, sőt folyamatosan növekszik. A szoftverkrízis[1] sok évtizedes fogalma és jelensége ma is húsbavágó, hatásait mindennapi életünkben valamennyien tapasztalhatjuk.

A mennyiség problémáján túl ebben komoly szerepe van annak, hogy a relációs adatbázisok kezelésének eszközei, a tervezési, fejlesztési módszerek és segédeszközök az elmélet lehetőségeinek, sőt követelményeinek szintjét nem érik el.

Ezen ellentmondásos helyzetnek számos oka és tényezője van a máig tartó elméleti vitáktól az üzleti szempontok túlzott térnyeréséig. Dolgozatomban a fogalmi szintű adatmodellezés és az ezen alapuló logikai és fizikai szintű tervezésnek a végtermék minőségére gyakorolt hatását vizsgálom. Egy kézzelfogható probléma modellezésén, megtervezésén és megvalósításán keresztül, a teszt-alkalmazás teljesítményének mérési eredményeivel támasztom alá azt, hogy a minőségi fogalmi szintű adatmodellezés bár önmagában nem elégséges, de megkerülhetetlenül szükséges előfeltétele a megfelelő minőségű végterméknek.

A technikai lehetőségek bűvöletében sajnos még ma is méltánytalanul háttérbe szorul a háromszintű modellezés-tervezés kérdésköre, nemcsak a szakirodalomban, de a fejlesztési módszerek területén is. Nincs olyan tervezőeszköz, nincs olyan tervezési-fejlesztési módszer, amely az elvárható szinten támogatná a fogalmi-logikai-fizikai szintű modellezést és tervezést. Ami ennél súlyosabb probléma: szakirodalma is alig van. Halassy dr. munkáin kívül nincs sem magyar, sem angol nyelvű irodalom sem, amely megfelelő elméleti alapozással tudományosan tárgyalná az adatmodellezés elméleti és gyakorlati problémáit.

A megfelelő fogalmi modellezési lépéssel kiegészített korábbi fejlesztési módszer alkalmazásával meglepően jó eredményeket lehet elérni, ami állításomat ha formálisan, matematikai értelemben véve nem is bizonyítja, de az erősen megalapozott sejtés szintjén valószínűsíti.

1 L. az Elvárások - szoftverkrízis c. részt, 121. old.

Alapfogalmak

„Új Bábelt élünk, a fogalmak pokoli zűrzavarát."
(Karinthy F.)

A mottónak választott Karinthy-idézet hellyel-közzel szaktudományokra is vonatkozik. Sajnos. Minden okfejtés, érvelés, eszmecsere alapja, sikerének szükséges - de nem elégséges - feltétele, hogy a felek az általuk használt elnevezések alatt pontosan ugyanazt értsék: azonos fogalmakat használjanak. Különösen és fokozottan igaz ez az informatikában, és különösen is vonatkozik az adatmodellezésre[2]. Sajnos területünknek nincs egységesen elfogadott magyar szaknyelve. Az még csak a kisebbik baj, hogy ugyanazt a fogalmat esetenként többféle névvel is illetik, bár a szaknyelvi pontosság és egyértelműség követelményével már ez sem fér össze. Nagyobb a baj, amikor nemcsak az elnevezésben vannak különbségek, sőt ellentmondások, amikor nyilvánvaló, hogy nem szinonimák kissé könnyelmű használatáról van szó, hanem bizony fogalmi zavarokról.

Különösen fájdalmas ez a jelenség éppen azon a területen, amelynek nemcsak nevében szerepel az információ, de amelynek elsődleges célja, hivatása az ismeretek célszerű átadása, közkinccsé tétele. „A tudományos nyelv nem abban különbözik a hétköznapi nyelvtől, hogy »másra« vonatkozik, hanem abban, hogy tömör, pontos, ellentmondásmentes, lehetővé teszi a szakemberek kommunikációját" [F4289 pp. 42-43.]

„Súlyos dilemmába kerültünk. Egyrészről ugyanis igaz, hogy védeni kell a nemzeti nyelvek tisztaságát és függetlenségét. Másrészt viszont az is igaz, hogy az új szakterminológiák »nemzetközi-angolságának« megőrzése bizonyos előnyt is rejt magában. Biztosítja mindenki számára a közös nyelvet, míg ha minden nemzet saját szaknyelvet kreál, azt csak a szakemberek és a szaktolmácsok értik." - írja Sebestyén professzor. [F4297 p. 33.] A pontos fogalmi meghatározottság szempontjából nem az az elsődleges probléma, hogy adott jelenséget milyen kifejezéssel nevezünk el, hanem az, hogy vajon ugyanazt a lényeget értjük-e alatta. Az igazi probléma nem az, hogy a bájt szó meghonosodása mennyire szerencsés vagy nem szerencsés magyar nyelvünkben, hanem az, ha a kilobájt alatt hol ezer, hol ezerhuszonnégy bájtot értenek...

Lássunk néhány kiragadott példát is, kivételesen forrásmegjelölés

2 L. még Fogalomalkotás, 140. old.

nélkül. Célom ugyanis evvel az, hogy magára a jelenségre rámutassak, és nem az, hogy az egyébként kiváló munkákat vagy szerzőiket lejárassam az esetlegesen kiragadott példákkal. Természetesen minden idézet pontosan dokumentált helyről, többnyire ezen munkában is hivatkozott művekből származik.

„Lényegében egy adatbázis nem más, mint hosszú ideig - gyakran évekig - meglévő információk gyűjteménye." [C4377] „Az adatbázis az információk jól strukturált, hasznos gyűjteménye." [C4457] Adat, vagy információ? Az egyébként alapos, jól használható, félezer oldalas könyv nem tér ki arra, hogy mi is az adat, mi az információ, és mi a különbség - ha van - közöttük.

„(Adat)egyed (entity): Minden olyan dolog (objektum), ami minden más dologtól (objektumtól) megkülönböztethető. Pl. személy, autó stb." [C4409] Egyedileg különböztethető meg, vagy fajtáját tekintve? Csak a példa segít eligazodni, az „objektum" fogalmát pedig nem határozza meg. „Az egyedeket tulajdonságokkal (attribútumokkal) írjuk le. Az autó egyedtípus esetén pl. típus, rendszám, szín, alvázszám stb." [C4409] Egyed vagy egyedtípus? A „típus" az kocsitípust jelent, vagy az egyedtípushoz lenne köze?

Az egyedtípus, egyed, egyedhalmaz, osztály; rekord, sor, egyedelőfordulás: igen gyakran rokonértelmű kifejezésekként használatosak, sajnos sok esetben következetlenül, nem pontosan definiált módon, nem azonos jelentéstartalommal.

A példák sorát igen hosszan lehetne folytatni.

Ennek a jelenségnek az okai lehetnek a szakterület fiatal volta, robbanásszerű bővülése, a hétköznapi életet közvetlen és egyre inkább meghatározó szerepe. Mondanivalóm pontos és világos megfogalmazása érdekében a legfontosabb alapfogalmakat megpróbálom ha nem is feltétlenül formálisan definiálni, de elég alaposan körüljárni ahhoz, hogy használatuk a továbbiakban egyértelmű legyen. Kezdetnek lássuk a - talán - legalapvetőbb és legfontosabb két fogalmat: az adat és az információ fogalmát.

Adat és információ

Adatbázisokról és információs rendszerekről szólván nem kerülhető meg az *adat* és az *információ* fogalmának tisztázása. A Magyar Értelmező Kéziszótár szerint:

„**adat** fn **1.** Vkinek, vminek a megismeréséhez, jellemzéséhez hozzásegítő (nyilvántartott) tény, részlet. *Gyártási, személyi ~ok.*" [F4313 p. 5.]

„**data** *lat* ismert tények, adatok, dolgok” [F4315 p. 159.]

„**data 1.** Factual information, especially information organized for analysis or used to reason or make decisions. **2.** *Computer Science* Numerical or other information represented in a form suitable for processing by computer. **3.** Values derived from scientific experiments.” (2. *számítástudomány* számítógépes feldolgozásra megfelelő formát képviselő számszerű v. más információ) [F4318]

„**információ** fn **2.** *sajtó* Értesülés, adat. **3.** *Tud* A kibernetikában: berendezésbe jelként betáplált adat; hír. [nk:lat] **~elmélet** fn *Tud* A kibernetikának az információk tárolásával és továbbításával foglalkozó ága.” [F4313 p. 593.]

Az adat magyar nyelvű általános, köznyelvi meghatározása ez esetben közel tökéletes szakmai szempontból is, ellentétben az amerikai angol változattal. Amit nem hangsúlyoz, ami nem nyilvánvaló, hogy a „valaminek” a megismerése nem az *adat* puszta begyűjtését jelenti, hanem a kapott adat a megismerés *lehetőségét* biztosítja. Ami még szükséges, az a kapott adat *értelmezése.* Ahhoz, hogy valami új felismerésre juthassunk újonnan megszerzett adatokból, azt értelmezni kell, meg kell ismerni, vagy fel kell ismerni annak *jelentését* az adott helyzetben[3]. Mondhatjuk tehát, ezt a jelentéstartalmi elemet hangsúlyozva, hogy az adat értelmezhető, de (még) nem értelmezett ismeret.

Az értelmezés nem más, mint a kapott új közlés által kiváltott gondolatsor, amely régebben megszerzett tudásunkon, tapasztalatainkon alapul. Ezekkel összevetve az új közlést, következtetést tudunk levonni, és az így megszerzett új ismeret - a kapott adatnak az adott helyzetben általunk tulajdonított jelentése - az információ.

Ezen megközelítés egyik következménye, hogy az információ meglehetősen szubjektív dolog. Ugyanazon adatból különböző helyzetben lévő különböző emberek más-más következtetést vonhatnak le, más-más jelentést tulajdoníthatnak az adott adatnak. Esetleg lehet olyan ember, aki semmilyen jelentést nem tulajdonít neki - akár a szükséges háttérismeretek, akár érdektelensége okán. Ez esetben nem biztos, hogy az adott közlés adatnak tekinthető.

Éppen ezért szempontunkból elsődleges fontosságú az adat jelentése, míg formája, tartalmi elrendezése bár nem érdektelen, de egyértelműen másodlagos. Az eredményes párbeszéd - kommunikáció - szükséges (de nem elégséges) feltétele, hogy a felek számára a fogalmak és azok jelentése egységes legyen.

3 L. még Ismeretszerzési folyamat, 11. old.

Az információ lényege nem a mennyiség, hanem a minőség [F4292 pp. 10-11.] „Az információ minősége összetett jellemző, amelyet különböző paraméterek együtteseként fejezhetünk ki. Ezek többsége nem számszerűsíthető (...) sokszor nehezen definiálható, viszonylagos, a minősítőtől erősen függő sajátosságról van szó.” [F4336 pp. 16-18.]. Ennek részletes ismertetése vagy taglalása a jelen dolgozat szempontjából nem elsődleges, arra viszont szeretnék rámutatni, hogy az adatbázisok egyik lehetséges felhasználási módja, a szelektív kezelés pont evvel van összefüggésben: olyan új ismeretek megszerzését segítheti elő, amelyeket más, hagyományos eszközökkel csak aránytalanul nehezen vagy egyáltalán nem lehetne megszerezni[4].

Ismeretszerzési folyamat

Nem kerülhetjük meg az ismeretek megszerzésének folyamatát, illetve ezen folyamat vizsgálatát. Az adatbázisok, illetve információs rendszerek használatának egyik lehetséges célja ugyanis éppen az, hogy új ismeretekre tegyünk szert segítségükkel. Halassy dr. szerint [F4260 pp.6-8.] az ismeretszerzési folyamatnak négy lépése van, az észlelés, az érzékelés, a felfogás és a megértés.

Az észlelés az ismerethordozó közeggel való időszerű szembesülés.

Az érzékelés lehet a köznapi értelemben vett érzékszervi érzékelés vagy valamilyen eszköz, berendezés által végzett fizikai, műszaki értelemben vett jelérzékelés. Pl. az ép szemű ember képes látni (valamilyen határok között), a számítógépben lévő hálózati (ethernet) kártya képes lehet a hozzá csatlakozó CAT-5-ös kábelen érkező elektromos jelek érzékelésére, kivéve például, ha a muzeális 10 Mbit/s sebességű kártyát próbálnánk meg 1 Gbit/s hálózaton használni.

A harmadik lépcsőfok a felfogás. Nem elegendő látni pl. egy „rajzot”, ismerni kell azt a jelkészletet, amelynek az adott „rajz” az egyik eleme. Például, ha az ismeretközlés papírra nyomtatott szöveg formájában történik, a fogadónak ismernie kell az ehhez használt jelkészletet, azaz betűket. Az ethernet kártya példáját továbbgondolva: nem elegendő az a tény, hogy a kártya és a hálózat egyformán 100 Mbit/s sebességű, óhatatlanul szükséges további feltétel, hogy a hálózaton szabványos ethernet-keretek közlekedjenek, különben a kártya nem tud mit kezdeni a jelekkel.

A negyedik lépcsőfok a megértés. Nem elegendő a jeleket ismerni, ismerni kell azt a nyelvet is, amelyen az üzenetet közlik. A számítás-

4 L. Ismeretszerzési módok, 45. old.

technikai példát továbbgondolva: szükséges egy olyan program, amely a kártya által „elfogott” keret tartalmával valamit tud kezdeni. Hiába küldünk egy adott IP-címre HTTP GET kérést, ha az adott gépen nincs webkiszolgáló program, amely valamit kezdeni tudna vele.

Ha mind a négy lépés sikeres volt, akkor mondhatjuk, hogy a fogadó számára rendelkezésre áll az az *adat,* amelyet a küldő vele tudatni szándékozott. Hogy ebből az adatból lesz-e információ, az már egy másik kérdés, azon múlik, hogy a fogadó azt tudja-e értelmezni[5].

Mindez azért fontos, mert az ismeretet időben, megfelelő, a fogadónak testreszabott formában, számára érthetően: érthető jelekkel és érthető nyelven kell közölni. Természetesen az ismeretet hordozó közeg sajátosságait, saját szabályait sem lehet figyelmen kívül hagyni. Mivel a legtöbb esetben az ismerethordozó közeg a nyelv, ezért különösen illik odafigyelni annak szabályaira, azok maradéktalan betartására is. Ha egy információs rendszer célja az, hogy ismereteket nyújtson, akkor azt ilyen módon kell tennie. (Régi élményem: a Keleti pályaudvaron szerette volna néhány külföldi, ha el tudom igazítani őket. Egyre azt hajtogatták, hogy „vekesszava”. Végül egy vasutas tudott rajtuk segíteni, tőle tudtam meg aztán, hogy bolgárok voltak, akik cirill betűs írásuk okán a B/b betűnket v hangnak olvasták...)

Rendszer

„**rendszer** fn **1.** *Fil Tud* Egynemű v. összetartozó dolgoknak, jelenségeknek bizonyos törvényszerűségeket mutató rendezett egésze.” [F4313 p. 1160.]

A rendszer fogalma nem választható el a modell fogalmától, ugyanis bármilyen rendszer esetében az adott „rendszer” meghatározása egyben a környezetétől való elhatárolását is jelenti, az pedig szempontfüggő, így egyben valamiféle modellalkotás is[6]. Előfordulhat ugyanis, hogy különböző szempontok, vizsgálódási (modellezési) cél esetében eltérő módokon határozzuk meg a rendszer szerkezetét, funkcióját, méretét, illetve határainak hollétét.

A legelső és a legfontosabb lépés a rendszer határainak rögzítése, azaz környezetétől való elkülönítése. Ez nem önkényes módon történik, hanem célszerűen, az adott vizsgálati szempontok szerint. A rendszer határainak rögzítése egyben megadja a vizsgálandó rendszer elhelyez-

5 L. Adat és információ, 9. old.

6 A modell hasonlósága a modellezetthez mindig valamilyen szempont szerinti. L. bővebben a Modell, modellezés, hasonlóság cím alatt a 16. oldalon.

kedését környezetéhez képest. A rendszer szerkezetét az alkotórészek (részrendszerek vagy alrendszerek, azok részei, egészen az elemeknek tekintett alkotókig) és a közöttük lévő viszonyok alkotják. A rendszer nem vizsgálható környezetéből kiszakítva, mert bárhogyan is állapítjuk meg határait, környezetével kapcsolatban van, a rendszer és környezete között állandóan több-kevesebb kölcsönhatás lép föl (nincs tökéletes szigetelés). A rendszer környezete tehát nem más, mint a rendszer komplementere. Raffai Mária meghatározása szerint a környezet „mindazon elemek, szabályok, normák, elvárások összessége, amelyek a rendszer hatáskörén kívül állva befolyásolják a rendszer működését...” [F4336 p. 29.] Ezen meghatározás inkább az informatikai vonatkozásokat hangsúlyozza mintsem mondjuk a fizikai értelemben vetteket, de evvel együtt is elfogadhatjuk akár általános érvényű meghatározásnak is.

Egyik szóhasználati mód szerint tárgyrendszernek nevezzük a vizsgált rendszert. Ehhez hozzárendelhető egy ún. célrendszer, amelynek állapotát az optimális, de legalább a biztonsági tartományon belül tartani a tárgyrendszer feladata. A tárgyrendszer kiszolgálója a célrendszernek, annak működését kell biztosítania. A célrendszer állapotai: optimális, biztonságos, átmeneti, tönkremeneteli. [F4289 pp. 156-157.] Azaz sosem szabad elfelejtkeznünk arról, hogy egy (az) információs rendszer feladata nem öncélú, nem önmagáért való, hanem valamely más rendszer (az adott célrendszer) optimális, de legalábbis még biztonságos működésének elősegítése, biztosítása. A követelménytartomány a tárgyrendszer azon állapotainak tartománya, amely mellett a célrendszer megfelelő (optimális, de legalábbis a biztonságos) működése biztosított.

Más szóhasználati mód szerint szokás (volt) a kiszolgálni kívánt - elsődleges - rendszert valós rendszer névvel illetni, míg az - azt kiszolgáló, másodlagosnak tartott - rendszert információs rendszernek nevezni. Nem mintha az adat, vagy akár az információ nem lenne valóságos...

Minden szervezetnek (vállalatnak) van egy - nem feltétlenül egyszerű - alapcélja, alapfunkciója. Egy kórház alapvető célja a beteg emberek meggyógyítása. Egy egyetem célja szakmailag (és emberileg!) kiművelt emberfőket képezni, nevelni. Ezen alapcélok elérését kell segítenie az információs rendszer megfelelő kialakításának, működtetésének, használatának.

Modell, modellezés, hasonlóság

„**modell** fn **4.** *Tud* Vmely jelenség, rendszer jellemzőit, összefüggéseit kifejező, ábrázoló, jelképező logikai v. matematikai formula, képlet.” [F4313 p. 963.]

Az ember „...a végtelenül sokoldalú tárgyak, jelenségek észlelésekor kiemeli (szelektálja) az adott körülmények között számára lényeges tulajdonságokat, elhanyagolva az észlelés szempontjai szerint lényegtelen elemeket. Ez az absztrakciós képesség - ami azt jelenti, hogy az ember agyában a valóság valamilyen modellje jelenik meg - teszi lehetővé az ismétlődések, az ok-okozati kapcsolatok felismerését.” [F4289 p. 39.] Ez az elvonatkoztatás, vagy modellalkotás teszi lehetővé általában az emberi fogalomalkotást. Ilyen modell például a fizikában a pontszerű test és a merev test fogalma. [F4290]

A modell kifejezés hallatán a legtöbb embernek a gyerekkori játék kisvasút és a „mecsbokszok” (matchbox) jutnak az eszébe. A modell, a modellezés azonban ennél jóval több és összetettebb dolog. A fenti játékvasút (modellvasút) ugyan valóban modellje az igazinak, a külső megjelenés, a forma tekintetében. A mecsboksz is modellje ily módon az igazi kocsinak. Mindkettő elég jól érzékelteti az eredeti *látványát.* De ha arra lennénk kíváncsiak, hogy az adott - mondjuk - Mercedes hogyan viselkedik, ha nekimegy a betonfalnak 120 km/h sebességgel, aligha használhatnánk ugyanezt a mecsbokszot vizsgálódásainkhoz. Vajon miért? Egyáltalán, miért van szükségünk modellekre?

Bármely kis részét is vesszük világegyetemünknek, az (gyakorlatilag) végtelen sok tulajdonsággal rendelkezik, az emberi megismerőképesség viszont véges. Ezen végtelen sok tulajdonság együttesét, egy reprezentációját *állapotnak* nevezzük. Mivel véges megismerőképességünkkel és véges technikai, anyagi lehetőségeinkkel sok esetben nem lehetséges a kívánt eredményt (gazdaságosan) elérni, ezért valamilyen módon *helyettesíteni* szeretnénk az eredeti tárgyat, technikai rendszert. (Vajon hány törékpróba kell egy biztonságosabb szerkezetű kocsi kialakításához? *Egy* kocsi mennyibe kerül?)

A tér végtelenségétől a végesre áttérve jutunk el a rendszer és a környezet fogalmához. Az idő végtelenségétől a végesre áttérve jutunk el a folyamat fogalmához. A tulajdonságok végtelenségétől a végesre áttérve jutunk el az állapotjellemzők fogalmához. Ezek segítségével a folyamatot is mint az állapotjellemzők változását írhatjuk le.

Mindig a vizsgálat célja (és természetesen tudásunk) szabja meg,

hogy mely tulajdonságok összességét tartjuk szükségesnek és elégségesnek az adott szempontok szerinti egyértelmű állapotjellemzéshez.

Az állapotjellemzés a vizsgálat céljától függ: az állapot maga objektív, a megfigyelőtől, annak céljától független, az állapotjellemzés viszont már szubjektív, függ a megfigyelő céljától, tudásszintjétől, a rendelkezésére álló eszközöktől stb.

Az állapotjellemzésnek egyértelműnek kell lennie: a vizsgálat szempontjától függ, hogy milyen állapotokat kell megkülönböztetni, és melyeket lehet azonosaknak venni, holott valójában nincs két azonos állapot. A tudomány - és a gyakorlat! - semmire sem jutott volna azonban, ha nem lett volna képes a feladatköre szempontjából egyértelmű állapotjellemzésre, ha nem lenne képes vizsgálatait (egyértelműen!) reprodukálni.

Az ismeretek szintjétől is függ az állapotjellemzés: nyilvánvaló, hisz csak azt vehetjük figyelembe, amiről tudomásunk van.

Az egyértelmű állapotjellemzéshez *szükséges* tulajdonságok összessége: a vizsgálódás céljától függ, hogy mely tulajdonságokat tartunk lényegesnek és melyeket lényegtelennek. A lényeges tulajdonságok közül sem szükségesek az adott szempontok szerinti vizsgálathoz azok, amelyek más - figyelembe vett - tulajdonságok függvényeként egyértelműen megadhatók.

Az egyértelmű állapotjellemzéshez *elegendő* tulajdonságok összessége: az elérendő cél eléréshez szükséges tulajdonságok közül egyet sem hagyhatunk el, mert ez esetben két megkülönböztetendő állapotot azonosnak tartanánk, ami viszont hibás következtetések levonását eredményezheti.

Az ilyen helyettesítést nevezzük modellalkotásnak. A modell általában bármi lehet, ami a funkciót biztosítja. Egy szemléletes megfogalmazás szerint a modell olyan póttárgy, amely az eredetit meghatározott célra és időtartamra helyettesíti [F4289 p. 182.]. A modellezés - az előbbiek szerint - valamely jelenség helyettesítése egy másikkal az eredeti jelenség tanulmányozása vagy bemutatása céljából. [F4291]

Ezt úgy is mondhatjuk, hogy a modell (csak) az adott szempontból hasonló a modellezetthez. Legyen például az a cél, hogy egy még csak tervezés alatt álló személykocsi karosszériájának légellenállási tényezőjét csökkentsük. Számításainkat ellenőriznünk kell, azaz a karosszériát kipróbálni a szélcsatornában. Ez esetben azonban a tényleges karosszéria helyett nyugodtan alkalmazhatunk egy azonos méretű, de jóval olcsóbban és könnyebben elkészíthető - geometriai - modellt, mondjuk hungarocellből. Ebben az esetben a pontosan ugyanolyan alakú és

ugyanakkora hungarocell jól helyettesíti az igazit, a vizsgálat szempontjából lényeges tulajdonságaik - méretük és alakjuk - azonosak.

Ha a cél például a törésállóság vizsgálata és növelése, akkor a modell egy számítógépes program lehet, amely kiszámítja, hogy az adott karosszéria az adott fajtájú ütközés során hogyan törik össze, közben mennyi energiát nyel el. Ha számításainkat végül ellenőriznünk kell, a szélcsatornából kivett hungarocellt aligha használhatjuk, hiszen a vizsgálat szempontjából az *nem hasonlít* az eredeti kocsira.

1. ábra. Patay László rajza

A modell hasonlósága a modellezetthez mindig valamilyen szempont szerinti. Másképpen jellemzi, más-más tulajdonságokat tart fontosnak - ugyanazokról az emberekről - például az adóhivatal és a modellügynökség, még akkor is, ha közöttük lehetnek olyan tulajdonságok, amelyeket mindkét vizsgálat fontosnak ítél. Teljes hasonlóság nincs. Olyan modell, amely *minden* szempontból hasonlít az eredetire, csak egyetlen egy van: maga a modellezett.

A modell, amit a fontos elemek kiemelésével és a kevésbé fontosaknak minősítettek elhanyagolásával kapunk, ugyanakkor jobban alkalmas a lényeg fel-, illetve megismerésére. Arany János: A walesi bárdok - ötszáz szó, alig háromezer betű, és mégis, micsoda kifejező erő! Patay László alábbi rajza is mindössze néhány vonal, és mennyire jellemző - szinte érezzük az öregember nehezedő mozgását.

A fogalomalkotás is egyfajta modellalkotás. Különféle jelenségek - számunkra fontos - közös jellemzőit megragadjuk, és kiemeljük a többi közül. Így jutunk el pl. a kocsi általános fogalmához, a különféle fajtájú, alakú, színű stb. konkrét példányok szemlélése után.

A fogalmakban kifejeződő modellalkotás

A nyelvi közléseknek van még egy sajátosságuk. Általános esetben az egyértelmű megérthetőségnek szükséges (de nem bizonyos, hogy elégséges) feltétele, hogy az ismeret mind a négy dimenzióját közöljük. Ez a négy dimenzió a generikus és specifikus alany és állítmány. A hétköznapi nyelvhasználat során ezek közül - látszólag - több is hiányozhat a közlésből, a nyelv sajátos szabályai, a szövegkörnyezet és a pillanatnyi helyzet alapján azonban azokat a felek többnyire egyértelműen be tudják helyettesíteni. Ha azonban az ismeretközlést formalizáljuk, szükséges, hogy a teljességre, evvel együtt az egyértelműségre törekedjünk. Így jutunk el (vagy így is eljutunk) az egyedtípus (ET), a tulajdonságtípus (TT), az egyedelőfordulás (EF) és a tulajdonságérték (TÉ) fogalmához.

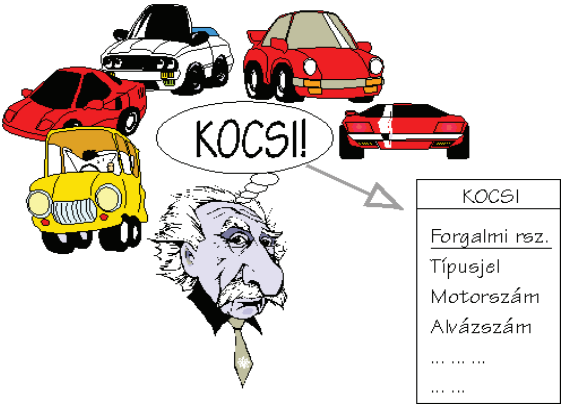
	alany	*állítmány*
generikus	(személy)	neve
specifikus	szomszéd	Lajos
generikus	ET	TT
specifikus	EF	TÉ

2. ábra. A generikus és specifikus alany és állítmány kapcsolata az egyedtípus, tulajdonságtípus, egyedelőfordulás és tulajdonságérték fogalmával

Egyednek hívunk bármit vagy bárkit, ami (aki) számunkra az adott vizsgálódás szempontjából lényeges. Ha számunkra - az adott vizsgálat szempontjából - lényeges, akkor kell legyen legalább egy, esetleg több olyan tulajdonsága, amely azt (őt) jellemzi. Az egyed tehát az, amit ismeretekkel írunk le, aminek (akinek) tulajdonságai vannak. „Az ismeretekkel leírandó jelenségek absztrakt osztályait egyedtípusokkal tükrözzük. A típusba sorolt konkrét egyed az egyedelőfordulás.” [F4260 pp. 26-27.]

Itt elvonatkoztatás, azaz modellalkotás jelenik meg. Első lépésben a konkrét egyedek - számunkra és az adott helyzetben - fontos jellemzőit

megragadva és kiemelve megalkotunk egy absztrakt, elvont osztályt, fogalmat teremtünk. Ennek az elvonatkoztatás útján nyert fogalomnak az informatika eszközeivel történő visszatükrözése az egyedtípus. Hasonló módon történik az elvonatkoztatás a tulajdonságtípus és a kapcsolattípus fogalmában.



3. ábra. Kettős modellalkotás az egyedtípus fogalmában

A modellalkotás szerepe

„Manapság, ha valakinek van egy jól tervezett adatbázis-kezelő rendszere, birtokolja az adatfeldolgozás és döntéstámogatás kulcsát."[7] [F4296 p. v.] A relációs adatbáziskezelés atyja, Codd így kezdi könyvének előszavát, amelyben összegyűjtötte a relációs elmélet kialakításával kapcsolatos írásait. Igaza van, mert a relációs megközelítés, illetve az ezen alapuló adatbáziskezelés olyan távlatokat nyitott meg az ismeretek kezelésében, amely korábban elképzelhetetlen volt, sőt, véleményem szerint még ma is messze vagyunk attól, hogy azt teljes mértékben kihasználjuk, hogy elmondhatnánk: eljutottunk a relációs megközelítés határáig. Szerintem ettől nagyon-nagyon messze vagyunk még *általában,* de egyedi példát sem tudnék mutatni (bár azt természetesen nem állíthatom, hogy ilyenek nincsenek).

Miért? Több tényező egymásra épülését, hatását kell itt vizsgálni.

7 „Today, if you have a well-designed database management system, you have the keys to the kingdom of data processing and decision support."

Amiről Codd beszél, az az általános célú adatbáziskezelő[8]. Az alapeszköz. Nyilvánvaló, hogy az eszköznek, a *szerszámnak* kifogástalannak kell lennie. Ez szükséges, de ismét nem elégséges feltétele a lehető legjobb eredmény elérésének. Ezen kívül - mint általában - döntő jelentőségű a *szerszámhasználat* módja, minősége. Azaz, esetünkben, hogyan is képezzük le a működő rendszert (célrendszer) az informatika síkjára, milyen adatbázist alakítunk ki.

Ez utóbbi terület három fő részre bontható: a modellalkotás és a tervezés, a kivitelezés (alkalmazásfejlesztés), és az üzemeltetés feladatköreire. Ezek a felsorolás sorrendjében épülnek egymásra. Ha a modell rossz, hibás, akkor a végeredmény sem lehet jó. Ugyanakkor egy jó modell alapján is lehet nem megfelelő módon, sőt akár nagyon rosszul végezni a kivitelezést, esetünkben az alkalmazásfejlesztést. Ha azonban a modellezés és a fejlesztés is (közel) tökéletes, még mindig hátra van az üzemeltetés feladata. Általános érvényű tapasztalat, amely természetesen érvényes az adatbázisok használatára is, hogy a legjobb eszközt is lehet rosszul használni, aminek következményei beláthatatlanok lehetnek.

Sajnos nincs sok okunk a bizakodásra. Soroljuk három csoportba az információs rendszer kialakításának költségeit: hardverköltség, szoftverköltség, szellemi munka. A '70-es években ezen három csoport aránya hozzávetőleg 60%, 30% és 10% volt. Húsz évvel később, a 90-es évek közepére az első két tényező aránya megfordult, de a lényeg, a harmadik a „szellemi munka" aránya maradt változatlan[9]. [F4314 p. 32.] Sajnos az azóta eltelt tíz év alatt nem történt érdemi változás. Az, hogy személyes tapasztalatból nem ismerek olyan információs rendszert, amely az optimális tartományban üzemelne, még magyarázható avval, hogy aránylag kevéssel találkozom közvetlenül. De az már gyanús, hogy széles ismeretségi körömben sincs példa erre. Sőt, nemhogy nem az optimális tartományban üzemelnek, hanem kifejezetten durva hibák fordulnak elő. Egy modellezési és egy alkalmazásbeli - elrettentő - példa következik.

Egyik eset (2007-ből): Nagyvárosi könyvtár, kölcsönzés és olvasóterem szolgáltatás. Minden kötetben vonalkód, az kölcsönzőjegy is vonalkódos. Beiratkozom csak helyben használatra. Ennek során az ügyintéző említi, hogy nekem ott már volt olvasójegyem. Igen, régebben kölcsönzésre jogosult tag voltam, olvasójegyem még meg is volt. Így az olvasójegy maradt is, nem kaptam új vonalkódot. Az olvasóteremben vé-

8 Az ehhez elvezető fejlődés logikai lépcsőfokait l. 40. old.
9 L. még az ún. 80-20%-os szabályt a 97. oldalon

gezvén szerettem volna kipróbálni, hogy a régebben kölcsönzésre is jogosító olvasójegyem vajon tényleg csak a helybeli használatra jogosít-e, és megpróbáltam kölcsönözni egy könyvet. Nem sikerült, mondta a könyvtáros, hogy evvel az olvasójeggyel sajnos nem lehet kölcsönözni, és el is tette a könyvet. Eddig rendben lenne. Két hónappal később azonban kaptam egy ajánlott levelet, miszerint haladéktalanul vigyem vissza a - szerintük - kölcsönzött könyvet (cím egyezik), különben bíróságon találom magam... Tehát jelzi ugyan, hogy kölcsönözni az adott jeggyel nem lehet, de azért mégis lekönyveli kölcsönzöttként. Nagy valószínűséggel tervezési hiba eredménye.

Másik eset (2009-ből): Felsőoktatásban használatos tanulmányi nyilvántartás. Reprodukálható módon előidézhető egy oldalon az a jelenség, hogy az egyik választási gombot megnyomva egy üzenet jelenik meg a képernyő közepén „Kérem várjon!" felirattal és egy „O.K." feliratú gombbal. Az „O.K." gomb megnyomása után az adott oldalon bármelyik gombot megnyomva az előző eset ismétlődik végtelen ciklusban, egyetlen lehetőség a kilépés. Megjegyzem, hogy a szóban forgó program a 377-es verziószámnál („Build") tart (2009. január 29-én).

Nem lehet eléggé hangsúlyozni a gondos tervezés fontosságát, ezen belül is a modellezés, a modellalkotás jelentőségét. Az adatmodellnek nemcsak jónak, de optimálisnak kell lennie[10]. A modellezésre fordított idő és energia bőségesen megtérül a későbbiekben. A számítástechnikában, jelesül a programozásban külön tanítják az ún. „frontális módszer" tarthatatlanságát, a minél gondosabb előzetes tervezés fontosságát hangsúlyozva. Sajnos nem sok eredménnyel.

Az információs rendszerek megtervezése (különösen az adatmodellezés), fejlesztése és üzemeltetése is meglehetősen összetett tevékenység. Mint ilyen, multi-, sőt interdiszciplináris ismereteket tételezne föl, jó elemző és problémamegoldási készséget igényel. Az adatmodellezés pedig - szaktudományi mivolta ellenére - olyan alkotókészséget, amely legalább egy kicsit a művészethez hasonlítható. [F4285]

Napjainkban jellemzően a technika kerül egyre inkább előtérbe ahelyett, hogy annak célszerű használati módja kapná a nagyobb hangsúlyt. A gondos és elmélyült modellezés hiányát többnyire a technikai eszközök egyre bővülő lehetőségei, szolgáltatásai próbálják ellensúlyozni, mérsékelt sikerrel. Codd a relációs adatbáziskezelés elvének kidolgozása során arra törekedett, hogy Einstein tanácsát kövesse: „Olyan egyszerű legyen, amennyire csak lehet, de annál egyszerűbb

10 L. bővebben: Az adatmodell jósága, 34. old.

ne". [F4363 p. vi.] Ez a szabály érvényes az adatmodellezésre is. A modell legyen olyannyira egyszerű, amennyire csak lehet, természetesen az elérendő cél figyelembe vételével[11].

A modellezés általában, az adatmodellezés specifikusan: *alkotó szellemi munka,* egy jó adatmodell bizonyos fokig tekinthető akár művészi jellegű teljesítménynek is. Természetesen az adatmodellezés is tanulható, gyakorolható, de szükséges hozzá valamelyes belső elhivatottság, készség is[12].

A relációs adatbáziskezelés elméletének kialakítása során Codd ellenállt a modell kiterjesztését célzó javaslatoknak. Úgy vélte, hogy nem helyes az általános érvényű modellt speciális elemekkel kiegészíteni, mert az fölöslegesen növelné a bonyolultságot: „A relációs modell minden felmerülő különleges igény szerinti kiegészítése helyett annak kommunikációs felületét kell jobban kihasználni a meglévő adattípuskészletet speciális meghívható eljárások fejlesztése által gazdagítva. Így a modell új, speciális elemekkel való bővítése elkerülhető."[13] [F4296 p. 447.]

Az ismétlődő adatcsoportok kapcsán Codd azt írja, hogy a relációs előtti rendszereken edződött embereket sokszor zavarta az, hogy a relációs modellben nem lehetnek ismétlődő adatcsoportok, és egy olyan példán keresztül mutatja be elkerülésük módját, amely máig iskolapéldának számít[14]. („Many people experienced with pre-relational systems express shock or dismay at the exclusion of repeating groups from the relational model. Repeating groups had their origin in the trailer cards of the punched-card era. Thus, it seems worthwhile to consider an example that illustrates how repeating groups can be avoided.") [F4296 pp. 30-31.]

Codd példája is azt támasztja alá, hogy a modellalkotás folyamata, a modell jósága[15] alapvető fontosságú. Ez számunkra ma is megfontolandó, követésre érdemes példa. A modellalkotás alkotó, teremtő tevékenység (benne van a nevében is!), ezért azt nem igazán lehet mennyiségi mutatókkal mérni, a modell jóságát pedig végképp nem. Ebből kö-

11 L. Patay László rajzát a 16. oldalon.
12 L. még: Elemzés és analízis, 79. old.
13 „Instead of expanding the relational model to handle every specialized need, the interfacing features of the relational model should be exploited so that the usual kinds of data handled by the model can be enriched by developing specialized invokable functions. It then becomes unnecessary to introduce new, specialized features into the model."
14 A több tételes megrendelések (iskola)példájáról van szó.
15 L. Az adatmodell jósága, 34. old.

vetkezően nem lehet normája sem, nem lehet általános szabályként kimondani, hogy mennyi idő alatt kell tudni egy modellt megalkotni. Hasonlóképpen a művészi alkotásokhoz.

Adatmodell

Mint láttuk a fentiekben, a valóság egy részét valamilyen sajátos cél elérése, vagy könnyebb elérése érdekében helyettesíthetjük valami mással. Fölmerül annak az igénye és/vagy lehetősége is, hogy *adatokkal* modellezzünk valamely jelenségeket, nem utolsósorban azért, mert egy megfelelő adathalmaz alkalmas lehet arra, hogy segítségével információkhoz jussunk a modellezett valóságról.

Ugyanakkor az is előfordul, hogy egy (információs) rendszer adatait, adatáramlásait, -tárolását, -feldolgozását szeretnénk modellezni, leképezni számítógépre. Akár így, akár úgy, az adatmodell fogalmának egy kézenfekvő meghatározásához jutunk el: A világ egyes részeinek *elemeire* nézve megnevezzük a fontos *tulajdonságokat,* adott szabályok szerint leírjuk *összefüggéseiket,* tartalmukat. Halassy Béla meghatározása szerint: „Az adatmodell véges számú egyedtípusnak, illetve azok egyenként is véges számú tulajdonság- és kapcsolattípusának a szervezett együttese." [F4292 p. 35.] Itt a többlet az egyébként kézenfekvő végesség hangsúlyozása, illetve a „szervezett" kitétel. Az adatmodell ugyanúgy rendszer, mint az általa modellezett valóság.

A rendszernek és határának kapcsán fentebb leírtakat[16] és az alapcélok, -funkciók[17] kiszolgálásának követelményét összevetve jogosan adódik az a következtetés, amit úgy lehet megfogalmazni, hogy az adott szervezeten belül „Az adatmodell - térben és időben - mindig egy. Az adatbázis több is lehet". [F4292 p. 74.] Peter Chen már 1977-ben foglalkozott evvel a problémával, mint olyannal, amit már régen - írja akkor! - meg kellett volna oldani: „Az elmúlt tíz évben az adatok logikai nézete jelentős figyelmet kapott. A legtöbb kutató azonban az adatok felhasználói nézetére összpontosította figyelmét. A vállalati adatnézet tanulmányozásának szükségességét a közelmúltig nem ismerték föl."[18] [F4298 pp. 77-84.] Chen ezen kívül azért is említést érdemel, mert neki

16 L. a 12. old.
17 L. a 13. old.
18 „The subject of the logical view of data has attracted considerable attention in the past ten years. However, most researchers have focused on the user view of data. The need for studying the enterprise view of data was not recognised until recently"

köszönhetjük a máig általánosan használatos ER-modellt (entity-relationship model), amit 1976-os cikkében mutatott be. [F4299 pp. 9-36.]

Chen ER-modellje nagyon érdekes dolog, nemcsak mert alapja a máig használatos adatmodellezési eljárások zömének, hanem azért is, mert rávilágít a „modell", „adatmodell", „(adat)modellezés" szavak lehetséges értelmezéseinek különbségeire. Codd az adatbázisok kezelésének egy - akkor - új modelljét alkotta meg, a relációsat, és évtizedek távlatából nézve is csodálatosan pontosan, a teljesség és a hibátlanság igényével alkotott maradandót. Chen ER-modellje *ebben az értelemben* nem modell, ahogyan azt Codd is megállapította:

„A relációs modellről szóló első két tanulmányom után körülbelül hat évvel Chen megjelentetett egy cikket az adatbáziskezelés egyed-kapcsolat alapú megközelítéséről. Ennek három alapvető problémája van: 1. csak a szerkezeti vonatkozásokat vizsgálta, sem az ezen szerkezeteken értelmezett operátorokról, sem a szerkezeti épség korlátairól nincs szó. Ennélfogva ez nem adatmodell. 2. Az egyedek és a kapcsolatok közötti különbség nem volt, és még mindig nincs pontosan meghatározva. Következésképp ami az egyik ember számára egyed, egy másik számára kapcsolat. 3. Még ha ez a különbséget pontosan meghatározta volna is, csak a bonyolultságot növelte volna többletteljesítmény nélkül."[19] [F4296 p. 7.]

Véleményem szerint Codd részben félreérthette Chen tanulmányát. A szóban forgó cikkét [F4316 pp. 9-36.] Chen ugyanis így kezdi: „Egy adatmodellt, az úgynevezett egyed-kapcsolat modellt mutatom be. Ez a modell egyesíti magában a *létező világ* fontos szemantikus információinak egy részét. Egy különleges diagramtechnikát is bemutatok mint az *adatbázistervezés eszközét.*"[20]

A dőlt betűs kifejezések alapján nyilvánvalónak látszik, hogy itt nem az adatbáziskezelés miként jéről (relációs vagy másmilyen) van szó, ha-

19 „About six years after my first two papers on the relational model [Codd 1969 and 1970], Chen [1976] published a technical paper describing the entity-relationship approach to database management. This approach is discussed in more detail in Chapter 30, which deals with proposed alternatives to the relational model. Although some favor the entity-relationship approach, it suffers from three fundamental problems: 1. Only the structural aspects were described; neither the operators upon these structures nor the integrity constraints were discussed. Therefore, it was not a data model. 2. The distinction between entities and relationships was not, and is still not, precisely defined. Consequently, one person's entity is another person's relationship. 3. Even if this distinction had been precisely defined, it would have added complexity without adding power."

20 „A data model, called the entity-relationship model, is proposed. This model incorporates some of the important semantic information about the *real world.* A special diagrammatic technique is introduced as a *tool for database design*."

nem adatbázisok tervezésről (azaz adatmodellezésről). Ugyanakkor rögtön a második bekezdésben Chen már arról ír, hogy az általa bemutatandó egyed-kapcsolat modell lehet az alapja a különféle „adatnézetek" (relációs, hálós stb.) egységesítésének. Véleményem szerint Chen alapos indokot szolgáltatott Coddnak a fentebb idézett véleményéhez. Evvel együtt is Chen munkásságát, illetve szóban forgó tanulmányát nem becsülhetjük le.

Emellett csak megerősíti azt a követelményt, hogy a használt fogalmakat nagyon pontosan kell meghatározni a tudományos és a műszaki életben (is!), mert ha nem ugyanarra gondolunk, amikor ugyanazt mondjuk, abból bizony nézeteltérés, rosszabb esetben teljes fejetlenség (káosz) lesz.

Az adatmodell rendszer is, mert elemekből és azok kapcsolataiból, viszonyaiból épül fel. Éppen ezért vonatkozik az adatmodellre is mindaz, amit a rendszerekről általános érvénnyel elmondhatunk. Az elemek és azok kapcsolatainak, viszonyainak megállapítása a modell szerkezetét adja. A szerkezet mellett nem szabad megfeledkezni a korlátokról sem, az adatmodellnek ugyanis van egy korlátvonzata is. A *korlátok* (constraint) az adatmodell tényezőire és azok viszonyaira vonatkozó megkötések, előírások halmaza (pl. a VÁLASZADÓ egyedtípus Életkor tulajdonságtípusa nem vehet föl negatív értéket; a hallgató csak olyan tárgy vizsgájára jelentkezhet, amelyet az adott félévben fölvett stb.). Ezek egymással is kapcsolatban állhatnak, tehát szintén rendszert alkotnak.

A modellezés során a modellezett valóságból kell kiindulni, azt kell tükrözni a vizsgálat céljának megfelelő módon[21]. Ezen a ponton a lényeg a „mit" és nem a „hogyan". Az adatmodellezés során nem kell, sőt nem szabad figyelembe venni a majdani megvalósítás esetleges eszközeit, azok tulajdonságait és főként korlátait[22]. Ez ugyanis eszközfüggőséghez vezet[23].

Az adatmodellt, az adatbázis modelljét nem szabad összekeverni az adatbáziskezelés modelljével sem. Az adatmodell szónak ugyanis lehet egy másik jelentése is, ami az adatok kezelési módjának általános alapelvét jelenti (relációs, hálós, hierarchikus, szemantikus [F4265 pp. 19-28.] stb.).

21 L. bővebben a Modell, modellezés, hasonlóság c. részben, a 15. oldalon.
22 L. bővebben A jelenlegi adatbáziskezelők elméleti korlátai c. részt, 59. old.
23 L. bővebben az Előtérben a technika, háttérben az elmélet, a modellalkotás és a szabványok c. részt, 67. old.

Az ún. százszázalékos elv[24] [F4269] szerint az adatokra vonatkozó minden ismeretet a (fogalmi szintű) adatmodellben kell rögzíteni[25]. Sosem szabad az adatszerkezetet, az adatmodellt egyszerűsíteni, ha ennek a kezelőprogramok bonyolultságának növekedése az ára[26].

Adatbázis

Adatbázis címszó még nincs a Magyar Értelmező Kéziszótár 1982-es kiadásában. Van azonban bázis címszó: „**bázis** fn **1.*** Alap. **2.*** Talapzat." [F4313 p. 95.]. Dacára annak, hogy a csillag a bázis szó ilyen értelmű használatát magyartalannak, sőt helytelennek jelzi, mégis segítségünkre van az összetett szó értelmezésében. Jelzi, hogy valami alapdologról, valaminek az alapjáról van szó[27].

„**adatbázis:** olyan adatok tetszőleges gyűjteménye, melyeket a rendszerint számítógéppel végrehajtott gyors megkeresés és visszanyerés céljából speciálisan rendeztek el. (...)" [F4323]

A Britannica Hungarica szócikke utalást sem tesz arra, hogy mit is jelent a „speciális" elrendezés, pontosabban azt sejteti, hogy ez a visszakeresés sebességének növelését eredményezi, annak érdekében szükséges. Semmilyen adatbázis nem lehet „adatok tetszőleges gyűjteménye", legfönnebb „tetszőleges adatok gyűjteménye", hiszen ha egy adatbázis egyáltalán megérdemli ezt a nevet, akkor igen komoly szerkezeti és egyéb megfontolások (adatmodell) vannak a hátterében.

Bármely adatbázist - figyelembe véve a célszerűséget - meg kell tervezni, a tervezés során viszont adatmodellt kell készíteni. Ezért az adatbázis fogalmának tisztázását valamilyen módon az adatmodellhez kell kötnünk. Ha a Britannica Hungarica meghatározását szeretném pontosítani, akkor azt valahogy így kezdeném el átfogalmazni: olyan ismeretek véges gyűjteménye, melyeket az adatmodellnek megfelelően rendeztek el. Tovább pontosítva: véges sok ismeretnek (egyedelőfordulásnak, tulajdonság- és kapcsolatértéknek) az adatmodell szerint és alapján szervezett együttese.

24 100% principle, ISO/IEC 14481
25 L. az adatszerű ismeretkezelési módok fejlődéséről írottakat, 40. old.
26 Mint pl. a tárgyi felmentések 8-as érdemjeggyel való jelzése, l. bővebben 107. old.
27 Vö. a kétféle ismeretszerzési móddal, 45. old.; valamint l. még a termelési és kutatási adatbázisokkal kapcsolatban elmondottakat, 45. old.

Háromszintű megközelítés

Fentebb az adat fogalmának kapcsán már szó volt arról, hogy a jelentés, a tartalom elsődleges, míg az adat megjelenési formája másodlagos[28]. Ennek azonban az adatmodellezésre kiható következményei is vannak: a modellezést-tervezést három különböző, de egymásra épülő szinten kell végeznünk. A hazai szakirodalomban ezt Halassy Béla dr. fejtette ki részletesen, másfél évtized alatt. [F4260] [F4314] [F4292] [F4276] [F4277] [F4278] [F4279]

Egy élelmiszercikk esetében nyilvánvalóan fontos adat a fogyaszthatóság határideje, olyannyira, hogy törvény teszi kötelezővé ezen adat feltüntetését az áru csomagolásán. Csakhogy ha a lejárat dátumaként azt látjuk, hogy „10-2-9", akkor lehet töprengeni a jelentésen: „2010. február 9.", „2009. október 2." vagy esetleg „2009. február 10.". Ebből is látszik, hogy bizony a forma nem hagyható figyelmen kívül, de még inkább következik belőle, hogy a tartalmat és a formát szét kell választani egymástól. A tartalom, a tartalmi szerkezet nem függhet attól, hogy pl. a majdani kezelőeszköz milyen dátumformátumot támogat (vagy nem támogat), arról nem is beszélve, hogy szükséges lehet esetenként az egyik, más esetekben a másik forma szerinti megjelenítés (pl. hazai piacra és exportra menendő cikkek esete). Ráadásul még semmit nem kell tudunk arról a modellalkotás fázisában, hogy a - majdani - kezelő milyen formában tárolja ezt az adatot. Egyazon tartalmi lényeg számos különféle formában ölthet testet[29].

Másik szempont az adatok esetleges logikai és/vagy fizikai partícionálása. Fölmerülhetnek olyan körülmények, amelyek szükségessé teszik az egyébként közvetlenül összetartozó adatok megbontását. Elképzelhető pl., hogy adott adatbázis-kezelő eszköz (RDBMS) hatékonysága csökken, ha egy adott egyedtípust jelképező táblának „túl sok" oszlopa vagy „túl sok" sora van. Ekkor lehet olyan megoldást választani, hogy a tulajdonságok (oszlopok) vagy az előfordulások (sorok) egy részét másik táblában helyezzük el. Nyilvánvaló dolog, hogy ez a körülmény nem hat vissza a modellezett valóságra, azaz az eredeti dolog attól még egységes marad minden tulajdonságával egyetemben. Ugyancsak nyilvánvaló, hogy a tartalmi lényeget az sem érinti visszaható módon, ha a

28 Vö. az adat jelentésének és formájának problémájával az Adat és információ c. részben, 10. old.

29 Ez persze adott esetben káros is lehet, tehát elkerülendő, erről szól pl. az írásszabvány, l. az Alkalmazási adatszabványok c. részt a 156. oldalon.

lehetséges előfordulásokat (sorokat) bármilyen megfontolásból (pl. földrajzi széttagoltság okán) több táblában tároljuk, amint hogy a Magyar Értelmező Kéziszótár [F4313] is *egy* könyv, annak dacára, hogy *két* kötetben jelent meg - a könnyebb kezelhetőség érdekében, a címszavak A-Ly és M-Zs felosztásával.

Ekkor nem beszéltünk még az esetleges hozzáférési korlátokról és követelményekről[30] sem. Ugyancsak nem részleteztem a különböző adatbáziskezelők adattárolási sajátosságait, sem az elemi adattípusok tárolásának, sem a kapcsolatok tényleges rögzítésének vonatkozásában.

Azt sem szabad figyelmen kívül hagyni, hogy általában több kiszolgáló rendszerünk van, az adott helyzetben alkalmazott adatbázis-kezelő rendszer ((R)DBMS) mellett, alatt megtalálható az operációs rendszer, esetleg valamilyen adatátviteli rendszer, az adatbázis üzemeltetését végzők ügyviteli rendszere stb.

Fogalmi modell

Összegezve: a tartalom elsődlegességének leszögezése mellett azonban a kép tovább árnyalható adatbázisok esetében (is). Ugyanis a tartalmi sík is két részre osztható, mégpedig a kezelők technikai megoldásai miatt. Ha egy adatbázis életében sor kerül egy kezelőváltásra, elképzelhető, hogy kevésbé vagy jobban, de meg kell változtatnunk adatbázisunk szerkezetét, holott eredeti mondanivalónk - az elsődleges tartalom - nem változott semmit.

Mindez szemléletesen alátámasztja annak jogosságát, hogy első megközelítésben ún. *fogalmi modellt* kell alkotni, azaz csak és kizárólag a modellezni kívánt valósággal szabad foglalkozni, azt kell a modellezésről általában elmondottak alapján és szerint úgy leképezni, hogy a kapott modell az adott cél szempontjából kellően hasonlítson a valóságra. A hasonlóság mértéke, vagyis a modell minősége nehezen mérhető, de a jó, sőt optimális modellel szemben támasztott követelmények világosan megfogalmazhatók[31].

A fogalmi modell a jelenségeket, azok sajátosságait és viszonyait a valóságnak megfelelően és természetes fogalmakban tükrözi. [F4260 p. 45.]

A fogalmi valósághűség követelményei közé az is beletartozik, hogy pontosan modellezzük az egyes adatok érvényesítési szempontjait, kor-

30 1992. évi LXIII. tv. a személyes adatok védelméről és a közérdekű adatok nyilvánosságáról [F4317]

31 L. Az adatmodell jósága c. részt, 34. old.

látait, azaz tartalmi összefüggéseiket. A fogalmi modellben kell rögzíteni minden ilyen jellegű ismeretet is.

Például egy hallgató adott tárgy vizsgájára jelentkezhet, ha a) aktív féléven van, b) az adott tárgyat fölvette, c) az adott tárgy követelménye vizsga, d) még nem vizsgázott belőle, e) az ismételt vizsgák engedélyezett darabszámát még nem érte el, f) ha már vizsgázott, akkor a vizsgaismétlési díjat befizette, g) van még kiírt vizsgaalkalom, h) a jelentkezési időkorláton még kívül van (előző nap délnél korábban), i) a vizsgaalkalomra eddig jelentkezettek létszáma nem éri el az engedélyezett legnagyobb létszámot... De sokkal egyszerűbb és kézenfekvőbb példát is lehet említeni: egy kérdőív feldolgozása során a válaszadó életkora: semmilyen körülmények között nem lehet negatív szám, a kérdőív jellegétől függően úgy kb. 6 év alatt és 90 fölött rákérdezünk, hogy biztos jó-e az életkor megadása, és kb. 120 év fölött ugyanúgy nem fogadjuk el, mint 0 alatt.

Az érvényesítési szempontok és korlátok pontos modellezése elengedhetetlen előfeltétele a későbbi eredményes üzemeltetésnek, ugyanis nagymértékben lecsökkenti annak esélyét, hogy hibás, rossz, ellentmondásos adatokat rögzítsenek a majdani ügyintézők.

Logikai terv

A fogalmi szintű modell valósághűségéhez ragaszkodva sem hagyhatjuk figyelmen kívül a (majdani) kezelő, általánosabban fogalmazva az üzemeltetési környezet technikai jellemzőit, korlátait, de ugyancsak itt említendők a hatékonysági és a hozzáférési (adatvédelmi) szempontok is, amelyekre fentebb láttunk példákat. Így jutunk el a következő szinthez, a logikai szinthez, és az adatbázis logikai szerkezetének nevezzük az a) technikai, b) hozzáférési, c) hatékonysági követelményeknek és korlátoknak megfelelően kialakított tartalmi adatszerkezetet. A fogalmi modellen több, egymástól eltérő logikai szintű terv is alapulhat. [F4260 p. 47.]

Fizikai terv

A harmadik vizsgálandó szint a fizikai szerkezet szintje. Evvel van a legkevesebb dolgunk, erre van a legkisebb ráhatásunk, ugyanis az adatbázis-kezelő ((R)DBMS) ezt alapvetően elrejti a szemünk elől. Ez egyrészt igen helyes körülmény, amelynek számos előnye van. Olyannyira, hogy már Codd megfogalmazta követelményként[32]. Másrészt

32 RS-1 The Information Feature [F4296 pp. 30-32.], RS-2 Freedom from Positional Concepts [F4296 p. 32.]

azonban lehet hátrány is, hiszen a felhasználó által tapasztalható sebesség és hatékonyság erősen függ az általában igen lassú háttértáron lévő fizikai adatok kezelési módjától. Ezen a szinten gyakorlatilag egyetlen eszközünk van: az indexelés, minden további megoldás (pl. szegmentálás) erősen eszközfüggő.

A fizikai adatfüggetlenség elve pont azt jelenti, hogy nem szükséges az elemi fizikai szintű műveletekkel foglalkoznunk az alkalmazás fejlesztése során, azokat az adatbáziskezelő végzi el helyettünk. Természetesen az általános célú kezelő hatékonysága nyilván nem éri el a speciális megoldások eseti hatékonyságát, viszont rengeteg munkát, következésképp rengeteg hibalehetőséget takarít meg.

Csak az arányok érzékeltetésére: az adatbáziskezelés alapműveleteinek számító keresések és rendezések hihetetlenül szerteágazóak és sokfélék lehetnek. Donald Knuth A számítógép-programozás művészete c. munkájának teljes harmadik kötetét[33] szánta erre a témára 761 oldal terjedelemben, és nem állítja, hogy a témát a teljesség igényével dolgozta volna föl.

A fizikai szintű adatszerkezetet az ismeretek tárolón való elhelyezésének, hozzáférésének és ábrázolásának (típus+méret) a tudatosan meghatározott rendjeként határozhatjuk meg. A fizikai szintű tényezők hatása azonban elsősorban a hatékonyságot befolyásolja. [F4287 p. 234.]

Az alkalmazott kezelőeszköz alapos - fizikai szintű - ismerete döntő jelentőségű lehet a hatékonyság szempontjából, ide értve annak SQL értelmezőjének működését is. Megtörténhet pl., hogy két, azonos listát eredményező SQL-lekérdezés futásideje között nagyságrendnyi különbség is van.

Leképezés

A fogalmi modellnek többféle leképezése lehet a logikai tervezés síkján, és ezen logikai szintű tervekhez ugyancsak többféle fizikai szintű terv tartozhat. Azaz a fogalmi-logikai-fizikai szintek hierarchikus, 1:N kapcsolatban állnak egymással. [F4260 p. 52.] A logikai szintű tervezés *nem* a fogalmi modell átalakítása, „elrontása”, hanem a fogalmi modell alapján történő tervezés, ezúttal már az alkalmazási (technikai, hatékonysági, hozzáférési) körülmények mérlegelése és figyelembe vétele mellett. Ugyancsak a fizikai szintű terv kialakítása *nem* a logikai szin-

33 The art of computer programming Volume 3 (Sorting and searching) - Addison-Wesley, 1973.; magyar fordítása: Keresés és rendezés - Műszaki Könyvkiadó, Budapest, 1988.

tű terv átalakítása, hanem az alkalmazandó technika sajátosságainak figyelembevételével megalkotott terv.

A három szint, a fogalmi, logikai és fizikai szint megkülönböztetése, továbbá a modellezés és tervezés ezen szinteknek megfelelő tagolása létfontosságú. Meggyőződésem szerint áll vagy bukik rajta a végeredmény termék minősége, esetleg akár használhatósága. Ezen túl számos előnye van. Ezek közül néhány: változások hatása, fizikai adatfüggetlenség, általános és egyedi nézetek.

Különös gondot kell fordítani a határesetek és határhelyzetek kezelésére. A három szintet alapul véve, két határhelyzet van, a fogalmi-logikai szintek és a logikai-fizikai szintek között. Az első eset a kevésbé problémás, ugyanis a fogalmi és a logikai szint világosan, jól elkülöníthető. A logikai és a fizikai szint határa a problémásabb, ugyanis a hatékonyság figyelembevételét logikai szintű tényezőként tartjuk számon, ugyanakkor a fizikai szintű megvalósítás igen komolyan visszahat erre.

Változások hatása

A legelső és talán legfontosabb előny, hogy három - különböző szintű - terv birtokában könnyebb szembenézni a valóság időnkénti óhatatlanul bekövetkező megváltozásával. Ha van világosan elkülönített három szintű tervünk, a változás hatása, illetve a változtatás igénye könnyebben, egyértelműbben és világosabban kezelhető. A változás ugyanis valamelyik szintet érinti a három közül, és elegendő az adott szintű tervet módosítani, és ezen következmények hatását továbbvezetni az alacsonyabb szintű tervben[34].

Fizikai adatfüggetlenség

Ha valamilyen oknál fogva szükségessé válik egy, a fizikai szintet érintő változtatás, az a logikai tervet elvben nem érinti, a fogalmi modellünket pedig végképp nem. Sőt, nemcsak a logikai szintű tervet nem kell megváltoztatnunk, de alkalmazásszintű programjainkat sem. A modellezett valóság ugyanis ugyanaz marad akkor is, ha az adatbázis-kezelő eszközünket frissítjük, vagy akár kicseréljük. Ezt úgy is nevezik, hogy fizikai adatfüggetlenség. Halassy Béla megfogalmazása szerint:

„Fizikai adatfüggetlenségről beszélünk akkor, ha az adatok elhelyezési, hozzáférési és ábrázolási módjában bekövetkező változáskor nincs szükség a programok módosítására.” [F4260 p. 50.]

Codd a relációs adatbáziskezelőkkel szemben támasztott követelmé-

34 Emlékezzünk pl. az ezredfordulós dátumproblémára, az Y2K problémára.

nyek között ezt úgy fogalmazza meg az általános tulajdonságok között (RS-4), hogy az adatbázis-kezelő teljes adattartalmának - ahogy azt a felhasználók látják - nem szabad függnie a kiszolgálótól vagy eszköztől, amelyben az adatok bármely része elhelyezkedik[35]. [F4296 p. 33.]

Sajnos a valóság nem feltétlenül ennyire tiszta és világos, ugyanis a „kompatibilitás", a „hordozhatóság", a „közös platform" stb. kifejezések és azok jelentéstartalma nem feltétlenül állják meg helyüket teljes mértékben, azaz a fizikai adatfüggetlenség elve sokszor nem érvényesül maradéktalanul, és ezért bizony igen sok esetben nem a fizikai szintű tervezés hiányosságai okolhatók. Szakirányú levelezési listákon a kérdések túlnyomó többsége valamely adatbázis-kezelő valamely verziójához kapcsolódik.

Általános és egyedi nézetek

Egy adatbázis elkészítésének csapatmunkájában részt vevő emberek három fő csoportba oszthatók. Van a vezető (menedzser), aki a megrendelő, és ilyen minőségében megfogalmazza a hivatalos elvárásokat. Van a fejlesztő (tervező), aki megpróbálja az elvárásokat megfelelően modellezni, majd arra a modellre felépíteni az adatbázist és az azt kezelő alkalmazásokat. A harmadik szereplő a tényleges felhasználó, akinek napi szinten kezelnie kell majd az alkalmazásokat. A három szereplő közül kettőnek általában nincs informatikai végzettsége, ami még nem lenne baj, de többnyire informatikai *műveltsége* sincs, ami viszont már baj. Ugyanis mind a vezető (a megrendelő pozíciójából) mind a felhasználó a saját *egyéni* elvárásait fogalmazza meg *kizárólagosként,* ráadásul a vezető az egész feladatot kizárólag projektként szemléli, és nem vesz tudomást arról, hogy vannak olyan részei, amelyek nem kezelhetők mennyiségi szemlélet alapján (pl. kétszer annyi ember esetén feleannyi idő). A fejlesztőt hajtja az idő- és gazdasági kényszer, és nincs ideje az érdemi, fogalmi szintű modellezésre...

Holott az egyéni - jogos - elvárások egységbe foglalhatók. Az adatmodell ugyanis szervezeti szinten egységes és egyetlen - kellene legyen[36]. Nincs ugyanis olyan felhasználási mód és cél, amely az egész adatbázist a maga teljességében, annak összes táblájával használná vagy használnia kellene. Olyan „felhasználó", akinek az egész adatmo-

35 „If a row of a base R-table is moved in any kind of storage by the DBMS, its information content as perceived by users remains unchanged, and therefore need not be changed. The entire information content of the DBMS as seen by users must not be dependent upon the site or equipment in which any of the data is located."

36 L. Adatmodell, 22. old.

dellt egységes egészként és egyben kell látnia, kettő van: a tervező-fejlesztő és az üzemeltető.

Ha azonban sikerül megalkotni egy egységes, jó[37] adatmodellt, akkor abból az összes egyéni, egyedi elvárás, nézet levezethető. Ezt egy hasonlattal tudom a legjobban érzékeltetni. Legyen adott egy atlasz a Kárpát-medencéről. Ebben van (lehet) sokféle térkép: domborzati, vízrajzi, közlekedési, közigazgatási, geotermikus, időjárási stb. Egyetlen fajta térkép nincs benne egész bizonyosan: olyan, amelyik egyszerre mutatja mind a felsoroltakat. Nincs is rá szükség. Viszont a felsorolt térképeket nem lehetne elkészíteni, ha nem létezne az általános (virtuális) térképi modellje a Kárpát-medencének. Ezen általános - és teljes - modell alapján azonban elkészíthetők az „egyéni kívánságoknak" megfelelő szakmai *nézetek.*

Hasonlóképpen: a jó adatmodell mint globális nézet alapján előállíthatók az egyéni, egyedi - parciális - nézetek a megfelelő igényeknek megfelelően, lehetővé téve a szűk szakmai igényeknek megfelelő, célszerű használatot.

Az ANSI-SPARC architektúra

Az, hogy három szinten kell szemlélni egy adatbázist, már nagyon régen kialakult. 1975-ben az ANSI[38] SPARC[39] bizottsága kiadta az ANSI-SPARC architektúra első változatát, amely már tartalmazza azt, hogy három szintet szükséges megkülönböztetni. Sajnos ez a három szint egész mást jelent ebben az esetben, mint a fentebb vázoltak, és nem is vált hivatalos szabvánnyá. Codd már többször idézett munkájában azt írja, hogy: „A bizottság egyik jelentésében alkalmazott három szint meghatározása szélsőségesen pontatlan, ezért számos módon értelmezhető."[40] [F4296 p. 33.] Ez a három szint a külső séma[41], a fogalmi séma[42] és a belső séma[43].

A Codd-féle relációs modellben is szerepel a három szintű megközelítési mód, sőt meg is felelteti az ANSI-SPARC szinteket a relációs modellbeli szinteknek. Fontos azonban tisztán látnunk, hogy függetlenül az ANSI-SPARC három szintjétől és függetlenül attól, hogy Codd erről

37 L. az Adatmodell jósága c. résszel, 34. old.
38 American National Standards Institute
39 Standards Planning And Requirements Committee
40 „The definitions of the three levels supplied by the committee in a report were extremely imprecise, and therefore could be interpreted in numerous ways."
41 external schema
42 conceptual schema
43 internal schema

hogy vélekedik, nem ugyanarról van szó! Codd a három szint alatt a felhasználói nézeteket, az alaprelációkat és a tárolási adatszerkezeteket érti[44] [F4296 p. 34.]

Az ANSI-SPARC, illetve Codd háromszintű architektúrája a relációs kezelőkkel szemben támasztott követelmények megfogalmazásának része, amiről viszont fentebb volt szó, az az adatmodellezés területe. Igaz ugyan, hogy a kétféle terület kétféle felosztása nem lehet teljesen független egymástól, hiszen - végülis - ugyanarról a valóságról szól mindkettő: a világ egy bizonyos részét, annak működését adatokkal szeretnénk leírni, és ilymódon jellemezni további ismeretek[45] megszerzésének, illetve a hatékonyabb működtetés[46] érdekében.

Nagyon fontos azonban, hogy tisztán lássuk a különbséget. Ezt nemcsak a fogalmi pontosság igénye követeli meg, hanem sokkal inkább az a körülmény, miszerint a fogalmi pontosság a továbblépés, az eredményesség előfeltétele. Így tehát mindenképpen muszáj különbséget tennünk az általános adatbáziskezelés relációs modellje (szemben mondjuk a hálós modellel) és annak tulajdonságai, valamint adott konkrét terület (pl. felsőoktatási intézmény tanulmányi ügyvitele) adatmodellje között.

A relációs kezelés modellje, illetve az ott alkalmazottak nem vihetők át egy az egyben a konkrét adatmodellezés területére. Ennek ellenére ezen keveredés számos változata a mai napig fel-felbukkan. „Az adatbázisok világában specifikálni kell egy szerkezetet, egy fogalmi vagy logikai sémát (DDL nyelven) (...) Különböző felhasználók (...) az adatbázisnak más-más fogalmi képét látják, részben azért, mert a menürendszerek alaposan elfedik az adatbázis igazi képét, részben pedig azért, mert a menüprogramok általában nem közvetlenül az adatbázishoz kapcsolódnak, hanem egy gazdanyelv + DML programhoz. (...) Az adatbázisnak az egyes felhasználó csoportok számára hozzáférhető részét nézetnek, zsargonban: vjú-nak (ang.: view) nevezzük. Lehet, hogy egy nézethez az adatbázis sémájának csak egy-egy alsémája (...) tartozik (...)” [F4362 p. 51.]

A „fogalmi kép” nem lehet különböző, nem keverhető össze a nézet („vjú”) fogalmával. Ha nincs világosan megfogalmazott globális fogalmi modellünk, amelyik ráadásul megfelel a következőekben részletezett jósági követelményeknek, nem tudunk - mert nem lehetséges - jó adat-

44 „RS-5, The Three Level Architecture: A relational DBMS has a three-level architecture consisting of views, base relations, and storage representations.”

45 Vö. szelektív mód, Ismeretszerzési módok és csoportosítások c. rész, 45. old.

46 Vö. az adatbázisok kétféle használat módjával, 45. old.

bázist létrehozni, amint a megfelelő globális térképi modell nélkül nem lehet elkészíteni a különféle szaktérképeket (vetület, „vjú")[47]. Létkérdés tehát a fogalmi, a logikai és a fizikai szint világos elhatárolása.

Az adatmodell jósága

Az adatmodellezés hasonlít egy kicsit a művészi alkotásra (is). Ahogy egy művészi alkotást „jobbnak" minősítünk, mint egy másikat, úgy az egyik adatmodell is lehet „jobb" mint egy másik. Míg a művészi alkotások minősítése meglehetősen egyéni, szubjektív, az adatmodellek mégiscsak tudományos-műszaki termékek (is), tehát szeretnénk objektív, tárgyilagos mércével mérni, minősíteni azokat.

Az adatmodell minőségéről, jóságáról, esetlegesen optimális voltáról méltatlanul kevés szó esik a szakirodalomban. Codd nem foglalkozik vele [F4296], igaz, ő általában véve sem a gyakorlati adatmodellezéssel foglakozik, hanem az *általános adatbáziskezelés* relációs modelljével. Ullman és Widom meg sem említik a kifejezést félezer oldalas - egyébként kitűnő - könyvükben[48] [F4263], pedig helye lenne benne.[49]

West és Fowler munkájukban nagyon helyesen rámutatnak a szabványok hiányára mint a „szegényes" modellezés okaira. A minőségi adatmodellezés feltételeiként hat szabályt neveznek meg, azonban ezek mindegyike technikai jellegű.[50] [F4319 p. 13.] Szó sem esik arról, hogy mitől jó az adatmodell - például attól, hogy *hasonlít* a valóságra.

Induljunk ki a rendszer és annak modellje(i) kapcsán fentebb leírtakból.[51] A modellnek *adott cél* szempontjából kell hasonlítani a valóságra. Minél inkább fennáll ez a hasonlóság, annál jobban közelíti a modell a valóságot. Ez az adott cél szempontjából szükséges és elégséges tulajdonságok precíz, pontos meghatározását jelenti, azaz mindazon vonatkozások meghatározását, amelyeket nem szabad figyelmen kívül hagyni. Tekintetbe véve, hogy számos dolog kölcsönösen meghatározhatja egymást, nem mindig könnyű azon tulajdonságok kiválasztása, amelyen nemcsak hogy szükségesek és elégségesek a modellalkotáshoz,

47 Vö. össze a térkép-hasolnattal, 32. old.
48 A könyv eredetije a Stanford Egyetemen jegyzet, magyar fordítását pedig több egyetemünkön, főiskolánkon használják tankönyvként (az MKM felsőoktatási tankönyvpályázatán támogatták fordítását).
49 Sajnos általános a jelenség, l. bővebben az Előtérben a technika, háttérben az elmélet, a modellalkotás és a szabványok c. részt az 67. oldalon.
50 Pl. minden egyednek kell legyen helyi(???) azonosítója az adatbázisban, és annak egyedinek kell lennie.
51 Modell, modellezés, hasonlóság, 14. old.

hanem ráadásul egyszerű, akár a lehető legegyszerűbb rendszert alkotják (vö. bázistranszformáció).

Ezen gondolatmenet alapján eljutunk odáig, hogy a jó adatmodellnek a) valósághűnek, b) teljesnek, c) minimálisnak kell lennie. Halassy Béla két további alapkövetelményt és két „ráadást" fogalmaz meg, [F4292 pp. 79-81.] avval a kiegészítéssel, hogy nem pusztán jó modellt, hanem a lehető legjobb modellt kell megalkotni.

Az öt optimumkritérium:

1) érthetőség
2) egyértelműség
3) valósághűség
4) teljesség
5) minimalitás

A két „ráadás" pedig a robusztusság és a bővíthetőség.

Az 1) és 2) sorszámú kitételek tulajdonképpen nyilvánvalóak. Miért kell ezeket külön említeni? A választ a történetiségben találjuk meg. A számítástechnika korai szakaszában a technika korlátai nagyon szűkre szabottak voltak, gondoljunk az olyan korlátokra, mint pl. fájl- és változónevekben megengedett karakterek, illetve ezen nevek megengedett hossza. Például a dBASE - ami ugyan nem adatbáziskezelő, hanem állománykezelő[52] - esetében ez max. 10 karakter - volt - és csak az angol abc nagybetűit tartalmazhatta[53]. A dBASE mintegy tíz évvel ezelőtt még jócskán „divatban volt". Ilyen előzmények után nem csoda, ha hangsúlyozni kell az érthetőséget. Nem véletlenül szerepel a fogalmi modell meghatározásában[54] is a „természetes fogalmakban" kitétel.

A fogalmi modell a már említett három résztvevő[55] közötti eszmecserék alapja és eredménye. Feltétlenül szükséges tehát ez a kitétel, és ennek maradéktalan betartása. Ez egyben előfeltétele a további követelmények vizsgálhatóságának is.

Az egyértelműség követelménye. Az adatmodellben nem szerepelhetnek sem természetes, sem technikai szinonimák és homonimák (hacsaknem nyomós okból, de akkor jól dokumentáltan). Ha az érthetőség és az egyértelműség követelménye mintegy előfeltételként teljesül, akkor vagyunk abban a helyzetben, hogy vizsgálhatjuk a valósághűség követelményét. Azaz mondhatjuk azt is, hogy az érthetőség és az egy-

52 L. részletesebben A fejlődés lépcsőfokai, 40. old.
53 Én már anekdotaként hallottam azt a leltározási történetet, miszerint csak igen hosszas fejtörés után jöttek rá, hogy a leltárban szereplő 100 m fókabél valójában egy jókora dob főkábel (FOKABEL).
54 L. 27. old.
55 A megrendelő, a fejlesztő és a felhasználó, l. 31. old.

értelműség követelménye olyan előfeltételek, amelyeket az adatmodellezés sajátos mivolta követel meg logikus és jogos többletként az általános értelemben vett modell jóságának nyilvánvaló követelményeihez képest.

Nem lehetnek modellünkben sem szerkezeti, sem tartalmi, sem pedig formai torzítások, mert ezek az adatmodell valósághűségét kérdőjelezik meg.

Tipikus szerkezeti torzítás pl., ha a hallgató nyelvtudását ilyeténképpen próbáljuk leírni HALLGATÓ(Hallgatóazonosító, Név, Nyelv, Nyelvvizsgafok, ...). Bár sajnos nem általános, de ha egy hallgatónak egynél több nyelvvizsgája van, akkor azt nem lehet nyilvántartani, azaz a mennyiségi viszonyokat torzítottan ábrázolja a szerkezet.

Tartalmi torzítás például az, amikor egy adott nevű tulajdonság tartalma valójában egész más, mint amire az elnevezés utal. Formai torzítások a különféle kódok - általános - alkalmazása a természetes megnevezések helyett.

A teljesség követelményét már a modellezés általános értelmezése is tartalmazza. Minden, az adott cél szempontjából lényeges tulajdonságra szükség van, a modellnek ezeket tartalmaznia kell. Ha lényeges elemek hiányoznak, akkor a modell nem lehet jó. Például ha egy tanulmányi nyilvántartásból hiányozna annak lehetősége, hogy az esetleges felmentéseket (testnevelésből mert élsportoló, idegen nyelvből mert megvan a nyelvvizsgája, adott szaktárgyból korábbi végzettsége alapján stb.) számon tartsák, az nagyon megnehezítené a használhatóságát. A teljesség követelményébe általánosabban az is beletartozik, hogy a modellnek kell tartalmaznia minden, az adatokkal és azok összefüggéseivel kapcsolatos körülményt, és sosem szabad az adatmodellt egyszerűsíteni a programbonyolultság növekedésének árán.

A minimalitás követelménye legalább két területen jelenik meg. Az egyik, hogy a szerkezetnek kell olyannak lennie, amelyik teljesíti az adatbáziskezelés talán legfontosabb szabályát, a redundanciamentességet. Nem elsősorban azért, mert nincs elegendő hely (sosem volt még ennyire olcsó a háttértár, mint manapság), hanem azért, mert ha egyazon ismeretet egynél többször tárolunk, akkor idő kérdése, hogy - karbantartási problémák miatt - az ismeret ellentmondásba kerüljön saját magával, például nem fogjuk tudni, hogy az ügyfél melyik címen is lakik a kettő (három, négy...) közül.[56]

56 A többszörös tárolásnak nemcsak szerkezeti okai lehetnek. Fővárosi kerületi önkormányzat nevét több mint 1100-féleképpen lehet leírni - sógorom, Karay Tivadar személyes gyűjtése. L. még az írásszabványokat: 156. old.

Ha pedig mégis sikerül kiküszöbölni (inkább: megúszni) a karbantartási anomáliák következtében fellépő adathibákat, az jelentős többletráfordítást követel anélkül, hogy az eredményesség garantálható lenne.[57] A redundanciának van ráadásul még egy kellemetlen következménye: a feldolgozások esetenként jelentősen megnövekedő időigénye.

A minimalitás másik területe, hogy az adott adatmodell határait a lehető legszűkebbre kell szabnunk, kerülve minden olyan vonatkozást, amely nem feltétlenül szükséges az eredetileg kitűzött feladat ellátásához.

A két „ráadás" követelmény esetében a robusztusság követelménye azt jelenti, hogy a modell olyan legyen, amely nagy adattömeget is képes megbízhatóan, gondok nélkül kezelni. A bővíthetőség pedig azt jelenti, hogy a modell bővítése esetén a már meglévő szerkezetet minél kevésbé, lehetőleg egyáltalán ne kelljen változtatni. Azaz törekedjünk az ésszerű mértékű általánosításra. Az érdekesség kedvéért megjegyzem, hogy ez utóbbi elemi programozási taktikai alapelv, a programozás középfokú oktatásában is régóta szerepel. [F4320 p. 30.]

Más szakirodalomban részben más feltételeket fogalmaznak meg az adatmodell jóságával kapcsolatban. Moody és társai az adatmodell minőségi jellemzőiként a teljességet, a szerkezeti épséget, a rugalmasságot, az érthetőséget, a helyességet, az egyszerűséget, az integráltságot és az alkalmazhatóságot követelik meg (ebben a sorrendben). [F4324 p. 22.]

Ezek részben megfeleltethetők a Halassy féle elvárásoknak. Az érthetőség nevében is azonos, a helyesség a valósághűség megfelelője lehet. A teljesség ismét nevében is azonos. Marad a szerkezeti épség, az egyszerűség, az integráltság és az alkalmazhatóság, ugyanakkor hiányzik az egyértelműség és a minimalitás. Az egyszerűség és a minimalitás ugyan rokoníthatók lennének, de fennáll az a probléma, hogy egy elegendően bonyolult rendszernek az adatmodellje sem lesz, nem lehet egyszerű. Ugyanakkor egy egyszerű modell is lehet redundáns, redundáns modell pedig nem lehet alkalmazható, ugyanis állandó karbantartási problémák merülnek föl a redundancia miatt. A szerkezeti épség jogos követelmény, de nem a modell jellemzője. A szerkezeti épség a megvalósult adatbázissal szembeni követelmény: pl. hogy a kulcs, vagy összetett kulcs esetén annak részei sem lehetnek üresek vagy ismeretlenek. A modell szerkezeti épsége akkor szenved csorbát, ha valamely

57 L. még az Üzemeltetői és adminisztrációs tapasztalatok c. részt, 106. old.

lényeges elemet nem tartalmaz (hiányzó kapcsolat), de ez ellentmondásban lenne a helyesség már megfogalmazott követelményével.

Az alkalmazhatóság érdekes követelmény. Ha egy modell nem alkalmazható, akkor ott súlyosabb problémák vannak: vagy maga a célkitűzés irreális a rendelkezésre álló lehetőségekhez képest, vagy pedig maga a modell kell kimondottan hibás legyen. Végül az integráltság követelménye semmitmondó, ugyanis minden rendszer - és az adatmodell is rendszernek tekintendő - egyben integrált is a kifejezésnek az „összetett, szervesen összekapcsolt" értelmében.

Úgy gondolom, hogy a Halassy féle öt optimum-alapkövetelmény fejezi ki pontosan a lényeget, közöttük nincs ellentmondás, nincs fölösleges (a többiből leszármaztatható), és kiegészíteni sem tudjuk igazán fontos követelménnyel - matematikai kifejezéssel élve bázist alkotnak.

A felsorolt követelmények közül a minimalitás az, amely a legkönnyebben vizsgálható, formális, matematikai (jellegű) eszközök alkalmazásával. Ezt az eljárást hívják normalizálásnak, amely a szerkezet sajátosságai alapján deríti föl a redundáns modellrészleteket, és teszi lehetővé ezek kiküszöbölését. A normalizálás folyamatát matematikai alapokon részletesen taglalja pl. Ullman és Widom már említett könyvében [F4263 pp. 138-188.].

A formális, „hagyományos" matematikai megközelítés mellett oktatási tapasztalatok alapján is keresnek újabb, szemléletesebb normalizálási eljárást, algoritmust, több-kevesebb sikerrel. [F4356 pp. 53-76.]

Halassy Béla szemantikus normalizálásról ír, a normalizálás egész folyamatát nem formális matematikai eszközökkel, hanem a modellezési elemek jelentését és azok összefüggéseit vizsgálva tárgyalja. [F4292 pp. 97-154.] Ennek alapján jut el a szemantikai normalizálás fogalmához, a modellalkotás valósághűségének követelményét hangsúlyozva a formális matematikai eszközök gépies alkalmazásával szemben, de azok szerepét nem lebecsülve.

„Inkább arra törekszünk, hogy a normalizálást a más szakkönyvekben leírtaknál mélyebben és átfogóbban értelmezve mutassuk be. Ezt azért kell megtennünk, mert az előnormalizálást nem lehet mechanikus módon végrehajtani. Eredetileg tervezett egyedeinkben az ismétlődések nem mindig láthatóan jelentkeznek. A mögöttes problémák ezért nagyon sokszor nem oldhatók fel tisztán matematikai eljárásokkal. A tervezőnek be kell vetnie szemantikai részmódszereket is. Például a valósághűség és az egyértelműség kritériuma jegyében elemi részekre kell bontania az összetett tulajdonságtípusokat, sőt, olykor a tulajdonságértékeket is. (...) Az előnormalizálás, a korrekt normalizálási alap

megteremtése összetett feladat. Azért az, mert ebben a fázisban sokkal nagyobb szerepet kap a tiszta józan gondolkodás, a helyes modellezési szemlélet, mint a sokkal könnyebben elsajátítható matematikai alapú mechanikus módszertár." [F4292 p. 97.]

A mértékadó szakirodalom szerint [F4321 pp. 364-365.] van jó és van rossz lebontás (dekompozíció, a kedvezőtlen függést tartalmazó egyedtípusok lebontása Úgy, hogy azokból a helytelenül függő tulajdonságtípusokat más - esetleg új - egyedtípusba visszük). Ez azonban hibás gondolatmenet. Nézzünk egy példát. A RENDELÉS(Rendelésszám,Vevőkód,Vevőnév) egyedtípus tranzitív függést tartalmaz. Formálisan szemlélve az egyedtípust és az átalakítást, kétféle eredményre juthatunk. Az egyik: VEVŐ(Vevőkód, Vevőnév) és RENDELÉS1(Rendelésszám, *Vevőkód*). A másik lehetőség: R1(Rendelésszám, Vevőkód) és R2(Rendelésszám, Vevőnév). Mindkettő veszteségmentes és - formálisan! - helyes megoldás is lehetne, mégis a másodikat rossznak tartjuk. Holott a formalitáson alapuló gondolatmenet a helytelen. Egyrészt két különböző egyedtípusnak nem lehet azonos a kulcsa. Másrészt a második „megoldás" karbantartási problémákat vet föl. Harmadrészt pedig, ha a *valóságot* próbáljuk modellezni, mindenképpen szükségünk lesz egy VEVŐ és egy RENDELÉS egyedtípusra, hiszen a modellezett valóság pont arról szól - példánkban - hogy a vevők rendeléseket adnak föl.

Szép példája annak, hogy a normalizálást nem lehet gépiesen, gondolkodás nélkül végezni. Ugyancsak példázza azt is, hogy ha a valóságot hűen modellezzük, akkor az efféle problémák fel sem merül(het)nek. Ez a megállapítás nem csökkenti a normalizálás formális matematikai megközelítésének értékét, ugyanis az segíthet föltárni a modell esetleges pontatlanságait, hibáit, amelyeket különben esetleg nem vennénk észre.

A modellezés egyben művészet is (egy kicsit), és mint ilyen, valamelyes képességet is feltételez. Amely képességet az igazi művészhez hasonlóan módszeres gyakorlással lehet, sőt kell is fejleszteni.

A relációs adatmodellezés kialakulása

Általában a történeti áttekintés elősegíti a jobb megértést, és nemcsak a történelmi, fejlődési folyamatokét, de a szakmai tartalomét is. Sokkal könnyebb megérteni a tudomány (és a gyakorlat) jelenlegi helyzetét, álláspontját, ha tisztában vagyunk vele, honnan is indult, hogyan jutott el idáig az adott szakterület.

A fejlődés lépcsőfokai

Az adatszerű ismeretkezelési módok fejlődése során három fő állapotot, fejlődési szintet különböztethetünk meg. Az első megoldás szerint az adatok tárolására használt fájl szerkezetét a kezelőprogramban definiáljuk, a kezelőprogram - tudván a fájl szerkezetét - annak megfelelően írja és/vagy olvassa azt. Ha eltérés van, akkor az adatok részben vagy teljesen összekeverednek.

Nézzünk egy Pascal példát:

```
type
      Adat=record
             Kulcs: LongInt;
             Megnevezes: String[64];
             Meret: Real;
           end;
var
   Egyadat : Adat;
   adatfile: file of Adat;
```

Először definiálunk egy összetett adattípust, rekordfajtát, amelynek három mezője van. Az első mező neve Kulcs, típusa négy bájton ábrázolt, előjel nélküli egész szám. A második mező neve: Megnevezés, típusa szöveges és maximum 64 karakter hosszúságú lehet. (Ez 65 bájton tárolódik, a string típusú változó 0. bájtja tartalmazza a tényleges hosszúságot.) A harmadik mező neve Meret, típusa lebegőpontos, valós szám. A változódeklaráció két változót deklarál, egy Egyadat nevűt, melynek típusa a fent definiált Adat nevű típus, azaz az Egyadat nevű változóban ilyen hárommezős rekordokat lehet elhelyezni. A második változó egy fájlváltozó, egy olyan fájlra lehet vele hivatkozni, amely Adat típusú rekordok egymásutánjából áll.

Ennek hátránya, hogy minden egyes esetben, ha változtatni kell az adatfájl szerkezetét - mondjuk kiderül, hogy nem elegendő a max. 64 karakteres szöveg a megnevezés céljára - nemcsak a programokat kell módosítani - a fájl szerkezetének leírása minden, az adott típusú fájlt kezelő programban szerepel! - hanem külön programot kell írni abból a célból, hogy a már létező fájl szerkezetét módosítsa, az adatokat konvertálja. (A régi szerkezetű fájlból egyenként be kell olvasni a rekordokat, mezőnkénti értékadással át kell tenni a beolvasott értékeket az új szerkezetnek megfelelő típusú rekordba, majd azt ki kell írni egy, a fentihez hasonlóan, de az új szerkezetnek megfelelően definiált fájlba.)

Ha sok efféle feladatunk van, kézenfekvő az a gondolat, miszerint az

adatfájlok szerkezetét nem az egyedi kezelőprogramokban kellene szerepeltetni, hanem magukban az adatfájlokban. Azaz csináljuk úgy, hogy minden adatfájl eleje tartalmaz egy fejlécet (header), amely valamilyen, de egységes módon megadja az utána következő adatok szerkezetét. Ekkor egy, általános használatú programmal tudjuk végezni az alapműveleteket (írás, olvasás, szerkezet módosítása) minden ilyen fájlon. Így jutunk el az állománykezelőkhöz (dBASE, Clipper, FoxPro és társaik)[58]:

```
DBASE - File header structure (DBASE III)

Offset Size            Description

  00   byte         dBASE vers num 03h=dBASE III w/o .DBT
                    83h=dBASE III w .DBT
  01   byte         year of last update
  02   byte         month of last update
  03   byte         day of last update
  04   dword        long int number of data records in file
  08   word         header structure length
  10   word         data record length
  12   20 bytes     version 1.0 reserved data space
  32-n 32 bytes ea. field descriptors  (see below)
  n+1  byte         0dH field terminator.


dBASE III Field Descriptors (FD count varies):

Offset Size            Description

  00   11 bytes   null terminated field name string
  11   byte       data type, Char/Num/Logical/Date/Memo
  12   dword      long int field data address, (set in me-
                                                    mory)
  16   byte       field length
  17   byte       number of decimal places
  18   14 bytes   version 1.00 reserved data area
```

58 Ezen eszközök szerepét a pc-s világban nem szabad lebecsülni: a fejlődés fontos lépcsőfokát jelentik, és még manapság is lehet látni használatban egy-egy, rájuk alapított nyilvántartást.

Ez komoly előrelépés az absztrakció szintjén, de még mindig hiányérzetünk van. Ebben az esetben az egyes fájlok kezelése az általános és egységes, egy darab kezelőprogrammal történhet, hiszen a különféle tartalmú fájlok szerkezetét minden esetben maga a fájl eleje tartalmazza. Ha azonban ilyen módon próbálnánk meg a relációsra hasonlító adatnyilvántartást csinálni, avval szembesülünk, hogy a táblák közötti kapcsolatok kezelése, továbbá minden korlát, érvényesítés kezelése a programozó „jóindulatára" van bízva. Vagy megcsinálja jól, vagy itt-ott téveszt - ember ő is.

Az absztrakció harmadik szintje az igazi adatbáziskezelés. Annak felismerése, hogy nemcsak az egyes fájlokat kell szerkezetüktől függetlenül egységes programmal kezelni, hanem a táblákat megszemélyesítő fájlok összességét, mégpedig a kapcsolatokkal, korlátokkal, érvényesítésekkel stb. együtt, az előre elkészített definíciónak megfelelően. Sajnos máig nincs olyan adatbáziskezelő, amely maradéktalanul megfelelne ezen elvárásnak.

Az állománykezelés szükségképpen procedurális jellegű, azaz számos lényegi elem - az egyes adatfájlok írásán-olvasásán túl - a kezelőprogramokba beépítetten van jelen. Evvel szemben az igazi adatbáziskezelés definitív, azaz minden, az adatokkal és azok összefüggéseivel, szerkezetével kapcsolatos ismeretet az adatbázis *definíciója* kell tartalmazzon. Ez olyannyira fontos követelmény, hogy külön neve van: ezt mondja ki az ún. százszázalékos elv.[59]

A kezdetek

A relációs adatbáziskezelés kezdete - mint oly sok találmányé - nehezen köthető egy adott naphoz. Ha mindenképpen meg akarjuk nevezni a kezdetét, 1969-re tehetnénk. Ekkor jelent meg a témában az első tanulmány, Edgar Frank Codd tollából „Derivability, Redundancy, and Consistency of Relations Stored in Large Data Banks"[60] címen. Ebben a cikkében a matematikai relációk adatbáziskezelésben való alkalmazhatóságát tárgyalja egy IBM kutatási beszámoló keretében. Codd 1970-es tanulmányát [F4271] szokták még emlegetni mint első publikációt.

Ebben a tanulmányban Codd hivatkozik D. L. Childs 1968-as tanulmányára,[61] akit tehát ötletadóként tarthatnánk számon.

Érdekes technikatörténeti kérdés lenne, hogy a relációs adatbáziske-

59 L. 25. old.
60 San Jose, IBM Research Report RJ599.
61 Description of a Set-Theoretic Data Structure

zelés kezdetét végülis mikortól számíthatjuk: az első olyan publikációtól, amely kifejezetten és részletesen erről szól (a lehetőség vagy a konkrétumok szintjén), vagy az első olyan *kezelő* megjelenésétől, amely az elmélet támasztotta követelményeknek legalább nagyjából megfelel. Mindenesetre Codd a kezdet kezdetén túlmegy a relációs adatbáziskezelés technikai lehetőségének fölvetésén, már ekkor hangsúlyozza pl. a szerkezeti épség megőrzésének fontosságát, pedig még nem is létezik relációs elvű adatbázis, sőt relációs adatbáziskezelő sem.

Codd ekkor 46 éves. Gyakorlatilag tökéletes alapozást csinál, a relációs adatbáziskezelést érdemi előzmények nélkül, szinte a semmiből teremti meg. Munkája értékét csak fokozza, hogy ekkor még az esetleges megvalósításhoz a technikai feltételek nem állnak rendelkezésre, nincs módja a gyakorlatban is kipróbálni elgondolását, kísérletezni.

Viszonyításképpen: ekkoriban (1969-70) még használható operációs rendszer is alig van. Ekkor még, illetve már dolgoznak a Bell Laborban a UNIX operációs rendszer ősváltozatán Brian Kernighan, Ken Thomson és Dennis Ritchie. Még nincsenek számítógépes hálózatok. 1973-ban jelenik meg az IBM 3340 lemezegység (erről kapta máig tartóan a „winchester" nevet a merevlemez), és kapott így a számítástechnika világa egy nagy lökést a további fejlődéshez. A (relációs elvű) adatbáziskezelésnek ugyanis nélkülözhetetlen technikai kelléke a blokkos tárolási elvű - azaz blokkonként címezhető - háttértár (mágneslemez).[62] Szalagos háttértárral gyakorlatban működő relációs elvű adatbáziskezelést csinálni ugyanis nem lehet.

Az IBM eleinte nem is alkalmazza Codd kutatási eredményeit. Tízéves küzdelmébe kerül, hogy elfogadtassa a szakmai közvéleménnyel. Könyvének ajánlása: „A Királyi Légierő II. világháborúbeli pilótáinak és legénységének és az oxfordi tanároknak. Tőlük ered elhatározásom, hogy harcoljak azért, amit igaznak hittem, azalatt a tíz-tizenvalahány esztendő alatt, amíg a kormány, az ipar és a kereskedelem erőteljesen ellenezte az adatbáziskezelés relációs elvű megközelítését."[63] [F4296 p. 6.]

Az általa lefektetett, sőt részletesen kidolgozott alapelvek máig érvényesek, egészen a lekérdezések kezelő általi optimalizálásával bezárólag. Természetes, hogy finomítások történtek az eltelt évtizedek so-

62 L. bővebben a Történeti kitekintés c. részt, 68. old.

63 „To fellow pilots and aircrew in the Royal Air Force during World War II and the dons at Oxford. These people were the source of my determination to fight for what I believed was right during the ten or more years in which government, industry, and commerce were strongly opposed to the relational approach to database management."

rán, sőt bővítési kísérletek is. Maga a lekérdező nyelv a hajdani SEQUEL-től az SQL mai állapotáig is komoly fejlődésen és szabványosítási lépéseken ment keresztül.[64]

A más szemléletű módok nem igazán terjedtek el, mondhatjuk, hogy máig egyeduralkodó a relációs elvű adatbáziskezelés. A megjelent publikációk alapján azonban a relációs elvű megközelítés az első, amely esetben az elmélet kidolgozása - teljes értékű módon - megelőzte a megvalósítási kísérleteket. Ebben az értelemben a relációs modell megelőzi a hálós és hierarchikus modellt.

Ez nem is csoda. Az első adatbáziskezelők ugyanis a gyakorlatias előzmények talaján bukkantak föl, a fájlkezelőkből alakultak ki. A fájlkezelés alapú rendszerek azonban nem felelnek meg az adatbáziskezeléssel szemben támasztott követelményeknek. [F4378 pp. 19-20.]

Adatbázis és információs rendszer

A hétköznapi szóhasználat az „adatbázis” és az „információs rendszer” kifejezéseket majdnem vagy teljesen rokonértelmű kifejezésekként használja, holott érdemes, mi több: szükséges különbséget tenni közöttük.

Az adatbázis fogalmát[65] már megvizsgáltuk. Ugyan az adatbázis fogalma nem szükségképpen kapcsolódik a számítógépekhez, de - a gyakorlatban legalábbis - nem lehetséges hatékony adatbáziskezelést végezni számítógép nélkül. Ezen értelmezésből az is következik, hogy egy adatbázis „önmagában véve” még kevés dologra jó, csak a lehetőséget teremti meg ahhoz, alapjául szolgál[66] annak, hogy ismereteket szerezzünk, illetve azokat célszerűen, hatékonyan kezeljük, feldolgozzuk.

Ennek megvalósításához szükségünk van különféle alkalmazásokra, amelyek az adatbázist használják, beleértve a karbantartást is, ez utóbbiak működtetéséhez pedig szükség van a megfelelő személyzetre, sőt a megfelelő ügyviteli rendszerre is. Ezek együttesét nevezhetjük információs rendszernek.[67] Raffai Mária a következő meghatározást adja: „Az információrendszer célja és feladata a valóság objektumainak, azok állapotának, viselkedésének és folyamatainak a jellemzése, elemeinek (információk, adatok) megbízható, pontos tárolása, ellenőrzése, rendszerezése, átalakítása, továbbítása, a szervezet célja szerinti

64 L. a Történeti kitekintés c. részben, 69. old.
65 L. 25. old.
66 Vö. a „bázis” idézett alapjelentésével, 25. old.
67 L. 40. old.

feldolgozása, új információk generálása és igény szerinti megjelenítése." [F4336 p. 57.]

Az adatbázis - akár a termelési, akár a kutatási mód[68] az elsődleges - erőforrás: mindenképpen pénzbe (időbe, energiába) kerül, és pénzt hozhat. Ennélfogva - mint minden erőforrással - gazdálkodni kell vele, korszerű kifejezéssel élve: menedzselni kell, különös tekintettel arra, hogy a tárolt adatok valósághűek, időszerűek, pontosak legyenek (ellenőrzés), a szükséges adatszolgáltatásokat a megfelelő időben, módon, formában teljesíteni lehessen.

Ismeretszerzési módok és csoportosítások

Egy adatbázis használata során kétféleképpen lehet ismeretekre szert tenni. [F4260 pp. 162-163.] Az egyik mód az ún. tallózás, ami ahhoz hasonlítható, mint amikor egy lexikonból kikeresünk valamit, akár célirányosan, akár véletlenszerűen, lapozgatva azt.

A másik lehetőség a szelektív mód. Ekkor az adatbázis tartalmából különféle, kombinált szempontok szerinti lekérdezések eredményeképpen juthatunk új ismerethez. Például kilistázzuk cégünk behajtatlan követeléseit nem azok névleges, hanem becsült értékeivel. A becsült értéket pedig olymódon állapítjuk meg, hogy az adatbázisban a behajtási kísérletek állomásainak tárolt adatait vesszük alapul (pl. jogerős ítélet esetében 60%-os, „címzett ismeretlen" esetben 5%-os stb. valószínűséget véve alapul), és a becsült értékek csökkenő sorrendjében listázzuk annak érdekében, hogy meghúzhassuk a határt a kiemelt, az átlagos és a leírandó adósok csoportjai között.

Ezt a megkülönböztetést Codd is említi. Ő nem ismeretszerzési módról beszél, hanem kétféle adatbázisról: a termelési és a kutatási adatbázisról. A termelésinek a célja szerinte az, hogy a kiszolgált rendszer (a vállalat) tevékenységeinek állapotát tükrözze bármely időben. A kutatási adatbázisé pedig az, hogy segítse megvizsgálni a többnyire jövőbeli lehetőségeket. Másképpen mondva a termelési a valóságot tükrözi, míg a kutatási a lehetőségeket. [F4296 p. 6.]

Codd megállapításával nem tudok maradéktalanul egyetérteni. Egyrészt nem az adatbázisok kétfélék, hanem a felhasználási mód, és az is változhat időről-időre. Egy „termelési" adatbázis tartalmazhatja (tartalmaznia kellene) mindazt az adatot, amelyek elemzésével pl. a vállalat átalakíthatja készletezési gyakorlatát úgy, hogy alacsonyabb átlagos készletszinttel is eleget tudjon tenni termelési tervének, evvel forgótő-

68 L. az ún. termelési és kutatási adatbázisokat alább.

két szabadítva föl. Ez a lehetőség ott van minden - jól tervezett - adatbázisban, a kérdés csak az, hogy annak üzemeltetője él-e vele, avagy sem.

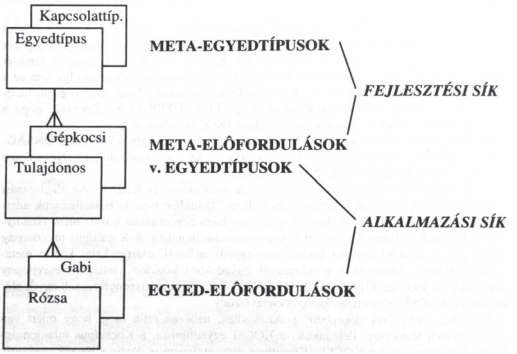
Ha egy ismeretet rögzített az adatbázisban annak üzemeltetője, akkor az a továbbiakban - nyilván - rendelkezésére áll. Bármikor, ha szükség van rá, elő lehet keresni, nemcsak önmagában, hanem sokféle - bármilyen - kombinációban és csoportosításban. Ezért célszerű a megszerzett ismeretet minél hamarabb rögzíteni az adatbázisban. Pontosan ugyanezért elegendő csak akkor előkeresni, amikor szükség van rá, mert az érdemi és értelmes használatnak a gondosan megtervezett adatmodell és az azon alapuló adatbázisszerkezet az alapja. Halassy ezt a „korai bemenet" és a „késleltetett kimenet" kifejezésekkel illeti. [F4260 pp. 117-118., 126.]

Az elsődleges cél az éppen szükséges ismeret hatékony megszerzését elősegíteni és kiszolgálni. Ez az elsődlegesen „termelési adatbázisokra" is igaz: az ügyfelek gyors, hibamentes, bürokráciamentes kiszolgálása a legfontosabb feladata.

A metaadatbázis szerepe

Az adatszótár minden fontos jellemzőt tartalmaz az adatbázisról, így az elemek pontos leírását: jellemzők, relációk, felhasználói struktúrák leírása (globális séma); felhasználó által definiált integritási feltételek; felhasználói igényeknek megfelelő adatstruktúrák (alsémák); felhasználói funkciók, feladatok; hozzáférési jogosultságok; a fizikai adatszervezés módja, adatbázis-adminisztráció stb. [F4335 pp. 115-116.] [F4266 7.10. fejezet]

Ennek szerepe pontosan az, amit a neve (szótár) is kifejez: az egységes és pontos, közös értelmezés biztosítása. Ez természetesen formalizált módon is kezelhető, sőt kezelendő. A kezdeti időszakokban ez természetes módon valamilyen állománykezelővel történt. A relációs adatbáziskezelés megjelenése után azonban logikus módon merült föl annak gondolata, hogy az adatszótár és az adatbázis sémája között igen jelentős átfedés van, ezt a kettősséget pedig érdemes lenne elkerülni. Így jutunk el a metaadatbázis fogalmához, amikor a fejlesztői és az alkalmazási sík tényezőit egységes rendszerbe foglalt módon - hiszen nem választhatók el egymástól - lehet kezelni. Halassy alábbi ábrája ezt jól érzékelteti. [F4260 p. 132.]



4. ábra
A fejlesztői és az alkalmazási sík egymástól elválaszthatatlan

Ismét a relációs adatbáziskezelés megkerülhetetlen megalapítójára, Coddra kell hivatkoznom. „A relációs elmélet egyik fontos eleme, hogy mind az adatbázis, mind annak meghatározása a felhasználók számára táblák együtteseként jelenik meg. Így - nagyon kevés kivételtől eltekintve - ugyanaz a relációs nyelv használható az adatbázis szerkezetének lekérdezéséhez és módosításához, mint magának az adatbázisnak a lekérdezéséhez és módosításához. Nincs szükség újabb betanulásra.” [F4296 p. 277.] Codd már korai munkáiban foglalkozik az adatszótár, illetve a metaadatbázis fontosságával, bár ekkor még nem ezen a néven említi azokat.[69] [F4296 p. 16.]

Ez teljesen összhangban van a már hivatkozott százszázalékos elvvel. Halassy megfogalmazása szerint: „Az adatmodell az ismeretekre vonatkozó összes ismeret tárháza, tehát a korlátokat is abban kell megadni.”[70] [F4260 p. 184]

69 Adatszótár - catalog; metaadatbázis - relational schema
70 Százszázalékos elv, l. 25. old.

2. Helyzetelemzés

A relációs adatmodellezés elméletének korlátai

Mind maga a relációs adatbáziskezelő eszköz, mind annak elméleti alapja „csak” *eszköze* az ismeretszerzésnek. Mint ilyen, nyilván nem lehet tökéletes minden szempontból. Ahogy a modellalkotással kapcsolatban általánosan bemutattam, a modell hasonlít ugyan - adott szempontok szerint - a valóságra, de nem azonos avval. Így igaz ez a relációs adatmodell esetében is. Ezért van az, hogy számos olyan kérdést lehet fölvetni, amelyre a relációs elmélet, illetve a relációs kezelők nem jó választ adnak, vagy nem jól adják a választ, vagy esetleg egyáltalán nem adnak választ. Ez azonban önmagában még nem baj. A baj az, ha nem tudatosítjuk egy (bármely!) modell alkalmazhatóságának határait, és ennélfogva egyes tulajdonságait hibának véljük. Ezek közül mutatnék be néhányat az alábbiakban.

Üres értékek

Egy adatbázisból kétféle módon hiányozhat ismeret. Vagy egész sor hiányzik egy táblából, azaz egy (vagy több) egyedelőfordulás nem szerepel, holott annak szerepelnie kellene. Ez az eset kívül esik a relációs elvű kezelés határain, modellezési-tervezési vagy üzemeltetési hiba eredménye lehet. („Az egyetlen mód, hogy ne hiányozzon a névsorból: ő olvassa.” [F4326 p. 157.])

A másik eset, amikor a sor a „helyén van”, csak egyes adatelemei hiányoznak, nem kap értéket minden tulajdonsága, másképpen mondva vannak üres cellák a sorban. Ez az eset különféle problémákat vet föl, amelyeket valamilyen módon kezelni kell.

Az adatmodell fogalmának tisztázása során leszögeztük, hogy „megnevezzük a fontos *tulajdonságokat* (...) leírjuk (...) tartalmukat”[71]. Ebből az következne, hogy minden tulajdonságnak minden esetben van értéke, holott ez nem biztos, hogy így van. Ha pontosabban végiggondoljuk a fenti fogalommeghatározást, akkor világossá válik, hogy nem erről van szó benne. A „leírjuk (...) tartalmukat” kitétel nem szükségképpen jelenti azt, hogy minden egyes esetben ismerjük az adott tulajdonság értékét, sőt még azt sem, hogy annak minden esetben lenne egyáltalán értéke.

71 L. 22. old.

Egy példával megvilágítva: egy háziorvosi nyilvántartásban elképzelhető olyan rovat, hogy a páciens (adott pillanatig, összesen) hányszor szült. Ez nyilvánvalóan egy nemnegatív egész számmal leírható ismeret. Értéke különböző esetekben lehet: a) pozitív szám, amikor tudjuk, hogy az adott hölgy hányszor is szült (pl. négyszer); b) nulla, amikor tudjuk, hogy az adott hölgy (még) nem szült egyszer sem (a szignifikáns nulla példája). Eddig problémamentes. Probléma azokkal a hölgyekkel van, akikről nem tudjuk, hogy hányszor szültek. Esetükben a megfelelő rovat üres marad (az inszignifikáns nulla esete), míg férfiak esetében szintén üres marad, de a két „üres" állapot nem egyenértékű. Férfiak esetében ugyanis a „szülések száma" nem értelmezhető („n.a."[72]). E két utóbbi esetet azonban mindenképpen szükséges megkülönböztetni egymástól, a valósághű modellezés[73] követelményén túl csak egyetlen példával érzékeltetve: ha születésszám-statisztikát kell készíteni, nyilván nem számolhatjuk bele nemcsak a férfiakat, de azon hölgyeket sem, akiknél az adat értéke - egyelőre - ismeretlen.

Erre a problémára született az a megoldási kísérlet, hogy az adatbáziskezelők egy különleges, NULL elemmel jelzik azt, ha az adott érték nem létezik. Ez mutatja azt, hogy az adott helyzetben az adat értéke ismeretlen, de nem különbözteti meg az „ismeretlen" esetét a „nem értelmezhető" esetétől, azaz két elvileg különböző esetet egybemos.

Ráadásul a hiányzó értékek problémája adattípusfüggő. Numerikus adatok esetében ugyanis a hiányzó érték és a „nem értelmezhető" nem különböztethető meg. Szöveges típusú adat esetében ugyanis van „üres" adat, az ún. üres string, amelyet olyan szövegkonstansként lehet megadni, amely semmit nem tartalmaz (insert into <táblanév> set <mezőnév>="";), azaz valóban az „ürességet" képviseli. Numerikus esetben azonban a nulla (0) az mindig szignifikáns. Így szöveges típusú adatelem esetében meglenne az „üres" („pillanatnyilag nem ismerem az adatot") jelentéstartalom is és a „nem értelmezhető" (NULL) jelentéstartalom is, viszont ugyanez numerikus adatelemek esetében így nem áll fenn, tehát a helyzetet súlyosbítjuk avval, hogy a jelenség függ az adatelemek típusától.

Ezen problémakör a kezdet kezdete óta fennáll, és máig nincs rá megnyugtató és egyértelműen világos megoldás. Ezt jelzi már az is, hogy Ullman és Widom azt írják kitűnő könyvükben, hogy „Ugyan a nullértékek nem képezik részét a hagyományos relációs modellnek, de

72 n.a. - not acceptable, és nem pedig „nincs adat".
73 L. bővebben az adatmodell jóságáról írottakat a 34. oldalon.

azért nagyon hasznos és kiemelkedő szerepet játszanak az SQL lekérdezőnyelvben". [F4263 p. 135]

Állításukkal ellentétben már az „alapító atya", Codd meglepően terjedelmesen és részletesen foglalkozik a problémával, idézett könyvében a teljes 8. fejezet[74] erről szól, míg a 9. fejezet az evvel kapcsolatos technikai kritikákkal foglalkozik[75]. Ezen kívül a 13. számú követelményben (RS-13) foglalkozik a hiányzó ismeret kezelésének módjával: „Azt a körülményt, ha egy érték nem áll rendelkezésre, az egész adatbázisban egységesen és módszeresen kell jelölni, függetlenül a hiányzó érték adattípusától. Erre a célra jelek szolgálnak."[76] [F4263 p. 39.] Azaz a hiányzó adat jelzése *nem érték*, hanem valamilyen más módon megvalósított *jelzés.*

A hiányzó, de az adott körülmények között értelmezhető eset jelölésére szolgál az ún. A-jel (applicable), míg a nem értelmezhetősége okán való hiányt az I-jel mutatja (inapplicable). A máig általánosan meglévő és használt NULL nem szerencsés megoldás és nem szerencsés elnevezés, mert valójában nem érték (ez az SQL szintaktikán is meglátszik, mert a 17 éveseket úgy kell lekérdezni, hogy „...where Kor=17", míg azokat, akiknek az életkora hiányzik, azokat nem úgy, hogy „...where Kor=NULL", hanem így: „...where kor is NULL"). Továbbá fennáll az a probléma, hogy két különböző esetet kellene megkülönböztetni az egyféle NULL használatával, ami nem megy.

Általánosabban fogalmazva, adatbázisok esetén a kétértékű logika nem elegendő. Az általánosan alkalmazott háromértékű logika is kevés, négyértékű logikára van szükség. Ezt Codd már 1986-87-ben, lassan negyedszázada leszögezte. A négyértékű logika implementálása nem lenne nehezebb vagy időigényesebb, mint a háromértékűé. Codd rámutat arra is, hogy ennek precíz megoldása fölöslegessé tenné a NULL használatát az SQL-ben. [F4263 p. 387.] A NULL kiküszöbölése pedig egyszerűbbé és egyértelműbbé tehetné a relációs adatbáziskezelés egyes területeit.

Date ugyancsak a NULL jel használata ellen érvel a relációs adatbáziskezelés kapcsán. [F4280 pp. 53-55.]

Lehetne azonban másképpen is érvelni, a valóság modellezésének oldaláról közelítve a problémához. Ebben az esetben viszont, a hasonló-

74 Chapter 8. Missing Information [F4296 pp. 169-196.]

75 Chapter 9. Response to Technical Criticisms Regarding Missing Information [F4296 pp. 197-206.]

76 Throughout the database, the fact that a database value is missing is represented in a uniform and systematic way, independent of the data type of the missing database value. Marks are used for this purpose."

ság követelményéből kiindulva megkockáztathatjuk, hogy nem is lenne szükség a „nem értelmezhető” jelentéstartalomra, amikor a NULL jel maradna az „adat még ismeretlen” állapot jelzésére. Ha ugyanis a valóságot modellezem, akkor kiragadom a számomra fontos jelenségeket, továbbá azok *tulajdonságait.* Egy létező jelenség létező tulajdonsága pedig nehezen elképzelhető, hogy értelmezhetetlen legyen. Másképpen fogalmazva, ha fölmerül adott helyzetben az - eseti - értelmezhetetlenség problémája, az valamilyen modellezésbeli pontatlanságra vezethető vissza.

Ez azonban egy olyan terület, ahol az elméleti teljesség és hibátlanság csak jelentősen nagyobb bonyolultság árán lenne elérhető. Célszerű tehát kisebb engedményt tenni az elméleti teljesség kárára a könnyebb és hatékonyabb modellezés érdekében, nem tévesztve szem elől azt, hogy az adott esetben értelmezhetetlen tulajdonságok gyakorisága a modell jóságának mértékére is utalhat.

Az üres/hiányzó értékek eddig vázolt problémája gyakorlatias eszközökkel kezelhető minden olyan esetben, ahol az múlhatatlanul szükséges, megfelelő további tulajdonságok beiktatásával. Vannak azonban további problémák is a NULL használata körül, és ezeket máig sem sikerült megnyugtató módon tisztázni. Márpedig nagyon kellemetlen, hogy egy közel negyven éves eszközben, amelynek szilárd matematikai alapjai vannak, ilyen tisztázatlan, nem egyértelmű, nem megnyugtató módon rendezett megoldások (illetve megoldatlan problémák) legyenek.

Igaz tehát, hogy a fentebb vázolt okfejtés alapján a hiányzó adatok problémaköre - elméletileg legalábbis - könnyen és gyorsan megoldható lenne. Mivel azonban ez a problémakör évtizedek óta jelen van, nem biztos, hogy összességében ez volna a leghatékonyabb megoldás. Már Codd fontos szempontnak tartja korai munkáiban is a kompatibilitás kérdését mint a felhasználók jogos és igen fontos érdekét. Így tehát marad a fennálló állapot - legalábbis még jó darabig - és az adott helyzet megkívánta módon orvosoljuk az elméleti probléma gyakorlati előfordulását technikai eszközökkel.

Date már 1975-ben példát mutatott arra, hogy a NULL használata esetén az SQL esetenként hibás eredményeket szolgáltat. [F4321 pp. 364-365.] Rubinson 2007-es cikke (NULL, háromértékű logika és kétértelműség az SQL-ben: Date kritikájának kritikája [F4312 36. évf. 4. szám pp. 13-17.]) vitába száll Date eredeti véleményével, mire válaszként megjelenik John Grant 2008-as cikke [F4312 37. évf. 3. szám pp. 23-25.] ugyanerről a kérdésről.

Date eredeti példája a következő: Legyen két tábla az adatbázisban: SZÁLLÍTÓ(sno, város) és ALKATRÉSZ(pno, város). Mindkét táblának egyetlen sora van, a SZÁLLÍTÓ esetében ez (s1,'London'), az ALKATRÉSZ esetében ez (p1,NULL), azaz a p1 azonosítójú alkatrész városa ismeretlen. Date kérdése: listázzuk azokat a sno-pno párokat, amelyek esetén a szállító és az alkatrész városa különbözik, vagy pedig az alkatrész városa nem Párizs. Az SQL ma megszokott formájában ez mint lekérdezés így nézhetne ki (az ékezetektől eltekintve):

```
select sno,pno from SZÁLLÍTÓ,ALKATRÉSZ »»
 »» where SZÁLLÍTÓ.város<>ALKATRÉSZ.város or  »»
 »»       ALKATRÉSZ.város<>'Párizs';
```

Az SQL válasza egy üres tábla. Date viszont azt mondja, hogy az (s1,p1) páros az elvárt válasz, és úgy érvel, hogy ha p1 ismeretlen városa Párizs lenne, akkor a feltétel második fele nyilván teljesül. Ha viszont p1 városa nem Párizs, akkor teljesül a feltétel első fele, ezeket a vagy logikai operátor kapcsolja össze, tehát a teljes feltétel a p1 bármilyen városa esetében igaz, így a válaszban benne kellene legyen az (s1,p1). De nincs. Pedig jogosan érezzük úgy, hogy a világ bármely városára igaz az az állítás, hogy az vagy Párizs, vagy nem az.

Egy másik, hasonló problémát mutat be Codd egyik példája. [F4263 p. 183.] Legyen egyetlen táblánk, amely alkalmazottak születési éveit (is) tartalmazza: ALKALMAZOTT(ano,...,születés_éve). Legyenek a következő évszámok a táblában: 1939, 1940, 1940, NULL.

```
select * from ALKALMAZOTT »»
 »» where születés_éve<1940 »»
 »» or születés_éve=1940 »»
 »» or születés_éve>1940;
```

Ebben az esetben a válasznak mind a négy sort tartalmaznia kellene, hiszen hiába ismeretlen a negyedik dolgozó születési éve, akkor is biztosra vehető, hogy ha az nem is pont 1940, akkor azt vagy megelőzi, vagy pedig nagyobb annál. Codd azt javasolja, hogy a jövőbeni adatbáziskezelőket fel kell készíteni az efféle egyszerű tautológiák felismerésére.

Véleményem szerint nem az a probléma, ha egy lekérdezésre nem - a saját logikánk szerint - elvárható választ kapjuk. Az első példabeli jelenségre ugyanis az a válasz, hogy azért van így, mert az ANSI így al-

kotta meg az SQL szabványt - dacára Codd, Date, Maier és mások munkásságának. Azazhogy a példabeli jelenség az SQL-nek nem hibája, hanem tulajdonsága. Az igazi probléma az, ha valamitől olyasmit várunk el, ami nem lenne feladata, ha megalapozatlanul - esetleg az értelmezési tartományon túlra - extrapolálunk.

Mindkét példa esetében igaz az, hogy a természetes észjárás alapján, pontosabban az egyik lehetséges természetes észjárás alapján elvárt választ *nem kapjuk meg*. A másik lehetséges megközelítési módja az efféle problémáknak ugyanis az, hogy az ismeretlen érték - éppen ismeretlen mivolta okán - nem vehet részt érdemi értékelésben. Mindkét példabeli esetben ugyanis az elvárt válasz feltételezné, hogy az SQL kérés feldolgozása során a kérdés *jelentésén alapuló értelmezést* végezzen az SQL parancsértelmező. Ez pedig - szerintem - messze nem feladata. Már csak azért sem, mert az említett példák aránylag egyszerűek. Ha elvárnánk az SQL-től, hogy az ilyesféle eseteket a példabeli módon kezelje, akkor igen komoly tartalmi-logikai elemzést (intelligenciát) kellene beleépíteni, hogy *minden* lekérdezés esetén végezze el az ehhez hasonló elemzéseket - hogy milyen eredménnyel, azt előre nem lehet megjósolni. Tehát jobb, ha inkább *következetesen* nem teszi.

A 'select' utasítás kiértékelésének érdekessége

Ullman és Widom példája egy egyszerűnek látszó halmazművelettel kapcsolatban. [F4263 pp. 281-283.] Legyen három táblánk, R, S és T, és mindegyiknek legyen egyetlen oszlopa, ezt jelöljük a-val. Listázzuk azon elemeket az R-ből, amely értékek megtalálhatók - az R-en kívül - vagy az S-ben, vagy a T-ben. Azaz az R metszet (S unió T) halmazra vagyunk kíváncsiak.

```
select R.a from R,S,T where R.a=S.a or R.a=T.a;
```

Abban az esetben, ha T üres, azaz egyetlen sora sincs, azt várnánk - halmazelméleti alapon -, hogy az R metszet S lesz az eredmény. A where feltételt nézve, az or logikai művelet igazságtáblája alapján pedig azt várnánk, hogy a feltétel igaz lesz minden olyan esetben, amikor R.a=S.a teljesül, függetlenül az or operátor másik oldalától. Ennek ellenére az üres halmazt kapjuk.

Ez esetben ez mindenképpen tulajdonság - még ha kicsit szokatlannak is tűnik -, ugyanis az RxSxT Descartes-szorzat T üres mivolta okán ugyancsak üres lesz, így az SQL nem tudja miből kiválogatni azokat a tételeket, amelyeket előzetes elvárásunk okán meg kellett volna találnia.

Tulajdonságok és egyedek nevei

Nem, vagy csak körülményesen lehet rákérdezni az egyedtípusok és tulajdonságtípusok neveire. Nem tudunk föltenni olyan jellegű kérdéseket az SQL-ben, hogy „Mely tulajdonságtípusok vehetik fel a 'zöld' értéket?” Ennek pedig aránylag kézenfekvően, könnyen kellene megoldhatónak lennie. Az adatbázist és annak definícióját, a metaadatbázist[77] ugyanazon kezelővel kellene kezelni, egységes egészként. Lévén adatokról szó mindkét esetben.

Codd szerint „A relációs modell egyik fontos tulajdonsága az, hogy mind az adatbázist, mind annak meghatározását - sémáját - a felhasználók relációk együtteseként észlelik. Így nagyon kevés kivételtől eltekintve ugyanazon relációs nyelv segítségével lehet lekérdezni és módosítani a az adatbázist, amivel annak meghatározását is. Nincs szükség új dolgok megtanulására.”[78] [F4296 p. 277.]

Halassy ugyanerről ír, de magyarázatot is fűz ahhoz, hogy ez máig miért nem valósult meg maradéktalanul: „A fejlesztői adatbázisok terén még ma is nagyon elterjedtek a testreszabott rendszerek. A fejlesztő számára adottak a metaadatbázis többnyire igen korlátos, a célszoftver képességeihez - és nem az igényekhez - szabott tényezői, és azok összefüggései. Nincs lehetőség a metaadatmodell változtatására.” „Talán azért, mert a szakemberek a végső-felhasználóknál is konokabban ragaszkodnak a megszokott megoldásokhoz.” [F4260 p. 135.]

Egyensúlytalan feldolgozás

Relációs kezelővel az alárendelt egyed tulajdonság-kombinációira vonatkozó lekérdezés csak aránytalanul nagy erőforrással oldható meg. Azaz ha nyilvántartjuk pl. emberek mindenkori lakcímeit, bármely ember eddigi (vagy bármikori) lakcímei viszonylag könnyen, pontosabban kevés erőforrás fölhasználásával kikereshetők, míg az afféle kérdéseket, hogy „kik azok, akik Kecskemétről költöztek Szegedre”, sokkal erőforrás-igényesebb megválaszolni.

Az első eset egy elemi lekérdezés:

```
select * from lakhely where ember=nn
```

A második eset kicsit összetettebb (feltételezzük, hogy a költözés

77 L. 46. old.

78 „An important property of the relational model is that both the database and its description are perceived by users as a collection of relations. Thus, with very few exceptions, the same relational language that is used to interrogate and modify the database can be used to interrogate and modify the database description. No new training is needed.”

során az előző cím megszűnésének napja egyben az új cím bejelentésének a napja is):

```
select L1.ember,L1.varos,L1.tol,L1.ig,L2.varos,L2.tol »»
   »» from lakhely as L1, lakhely as L2 where »»
   »» L1.ember=L2.ember and L1.varos="Kecskemét" »»
   »» and L2.varos="Szeged" and L1.ig=L2.tol;
```

Ekkor ugyanis két sorváltozóra lesz szükségünk, mivel egy reláció sorait ugyanazon reláció soraival kell összehasonlítani. Ennek kiértékelése pedig a következő módon történik [F4263 p. 281.] nyomán:

```
MINDEN t1 sorra az L1 relációban
   MINDEN t2 sorra az L2 relációban
      HA a where záradék igaz, amikor az attribútumokban
         t1 és t2 megfelelő értékei találhatóak, AKKOR
         t1, t2-nek megfelelően kiértékeljük a select
         záradék attribútumait és az értékekből alkotott
         sort az eredményhez adjuk
   CIKLUS VÉGE
CIKLUS VÉGE
```

Vagyis annyiszoros Descartes-szorzatot képez a kezelő, ahány összehasonlítást kell tenni az adott reláció sorai között. Ennek számossága a sorok darabszámának annyiadik hatványa, amennyi az összehasonlítások száma, példánkban kettő. Az első esetben tehát a sorok (tárolt lakcímek) számával lesz egyenlő, a második esetben pedig ezen szám négyzetével. Ha netán városhármast keresnénk, akkor a harmadik hatványával. Ha a szóban forgó tábla számossága nagy, eléri mondjuk a néhány (tíz)millió címváltozást, valamilyen módon könnyítenünk kell az adatbáziskezelő dolgát (mondjuk az összes cím közül a kérdéses városokat tartalmazókat ideiglenes táblába gyűjteni, és azon a jóval alacsonyabb számosságú táblán végezni a lekérdezést.), de ez már logikai, illetve fizikai szintű eszköz alkalmazását jelenti.

Matematika-függőség

A relációs adatbáziskezelés szilárd matematikai alapokon nyugszik. Ez azonban nem korlátozó körülmény, hanem inkább előny, még akkor

is, ha néhány esetben nem várt eredményre vezet.[79] Codd az adatbáziskezelők tervezési alapelveiről szóló fejezetben legelsőnek veszi azt a szabályt, miszerint egy kezelőeszköz vagy annak lekérdező nyelve(i) nem sérthetik a matematika szabályait. [F4296 p. 351.] Tekintetbe véve azt, hogy a matematika a világ tudományos-műszaki leírásának kiváló (segéd)eszköze, csak helyeselni lehet ezt a körülményt.

Esetleges ellentmondás felfedezése esetén pedig kiváló lehetőség az elgondolkodásra, a hiba vagy az ellentmondás helyének megkeresésére; ellenkezőleg pedig, amikor pont az összhangot tapasztaljuk meg a matematikai megközelítés és a modellezési megközelítés között, megerősít bennünket a modellalkotás helyességében. Például: a reláció halmazelméleti meghatározása kizárná, hogy egy táblában két azonos sor fordulhasson elő. Ennek ellenére ezt egyes adatbáziskezelők egyes esetekben megengedik - könnyítést tesznek a formális matematikai szabályok érvényesülésének kárára. Kérdés, hogy van-e ennek értelme, haszna?

Nézzük az ismétlődő sorok problémáját modellezési szempontból! A modellalkotásról[80] fentebb elmondottak alapján a modellalkotás (adatmodell-alkotás) lényege, hogy megállapítjuk az összes fontos tulajdonságot, és csak a fontosakat.[81] Azon esetek, amelyek a fontosnak tartott tulajdonságaik *mindegyikében* megegyeznek, azonosnak tartandók. Azaz, ha két sor azonos ugyanazon táblában, az azt jelentené, hogy a két sor által reprezentált két egyedelőfordulás azonos, azaz a valóság ugyanazon elemét írják le. Ezt pedig nyilván elegendő egyszer megtenni, sőt nemcsak fölösleges, de káros is egynél többször megtenni. Ha van két ember, akinek a neve, anyja neve, születési helye és ideje azonos, akkor az a két ember (eddigi történelmi tapasztalatok szerint) azonos, valójában ugyanazon személyről van szó. Ha azonban „csak" a leíró tulajdonságok azonosak, de a (mesterséges) azonosító értéke különböző, akkor viszont az a kérdés merülhet föl egyes esetekben, hogy vajon jó-e a modellalkotás (teljesség követelménye), azaz kiterjed-e minden lényeges tulajdonságra?

Kiegészítés: axiómaként mondjuk ki, hogy minden egyedtípusnak kell legyen azonosítója,[82] és annak értéke egyetlen előfordulás esetén sem lehet üres vagy ismeretlen. Ebből ugyancsak az következik, hogy

79 Vö. üres értékek problémaköre, 48. old.
80 L. 14. old.
81 Azaz teljes és minimális adatmodellt alkotunk. L. Az adatmodell jósága, 34. old.
82 L. Codd álláspontját, 65. ol.d

egy egyedtípusban nem lehet két azonos előfordulás, azaz az ezt megvalósító táblának nem lehet két azonos sora.

Csoportok és atomi értékek

A relációs adatbáziskezelésben - elvileg - nincsenek összetett adattípusok, minden adattípus elemi, vagy másképpen mondva atomi, azaz az adatbáziskezelő azt nem tudja kisebb részekre bontani (leszámítva esetleg egyes beépített függvények használatának esetét). Ez alól egyetlen kivétel van: maga a reláció, viszont ebben az értelemben azt nem tekintjük adat*típus*nak. Az összetett adatok az elemiek különféle kombinációival írhatók le. Sem a relációs modell, sem az ilyen elvű kezelők többsége nem támogatja a csoportok alkalmazását.

Codd eredeti érvelése evvel kapcsolatban az, hogy minden összetett adattípus alkalmazása fölöslegesen növelné a bonyolultságot anélkül, hogy érdemi többletteljesítményt nyújtana. Ugyanis a kezelőnyelv(ek)-ben legalább négy alapparancsot kell alkalmazni (lekérdezés, módosítás, bevitel, törlés), és ez azt jelenti, hogy minden egyes további (összetett) adattípus esetén legalább néggyel növekszik a parancsok száma. Márpedig az eredeti célkitűzés[83] az volt, hogy az elmélet és a kezelők a lehető legegyszerűbbek legyenek.

Ullman és Widom is az egyszerűségre hivatkoznak evvel kapcsolatban és az ennek tulajdonítható elterjedtségre: „Ezzel szemben az az igazság, hogy a relációs modellen alapuló adatbázisrendszerek a legelterjedtebbek a piacon. Ennek az egyik oka, hogy a modell egyszerűsége miatt elegáns és hatékony programozási nyelvekkel lehetséges az adatbázisok lekérdezése.” [F4263 p. 114.]

Ugyanakkor jogos az igény az adatcsoportok - mint például a dátum - alkalmazhatóságára. Egy dátum általános esetben három elemi adat *csoportja:* év, hónap és nap alkotja. A nem-relációs - pl. az objektumorientált - megközelítésben viszont minden további nélkül lehetnek összetett adattípusok, ennek összes előnyével - és hátrányával. [F4262]

Az adatcsoport fogalma, illetve annak alkalmazása nélkül nem lehet megvalósítani az optimális adatmodellt, azaz nem lehet teljes értékű normalizálást végezni (l. Armstrong axiómáját: egy összetett tulajdonság mindig meghatározza a saját összetevőit [F4261 pp. 580-583.]). Vegyük észre, hogy a relációs modell, illetve a relációs kezelők annyiban nem következetesek, hogy legalább egy esetben támogatják a csoport alkalmazását: ez pedig az összetett kulcs esete.

83 Codd e tekintetben Einsteinre hivatkozik, l. 20. old.

Altípusok

Az altípusok fogalma, szükségessége igen korán fölmerült, Codd 1979-ben már részletesen tárgyalja. [F4327 pp. 410-411.] Az altípus fogalmának kapcsán tárgyalja a generalizáció és specializáció eseteit is. [F4327 pp. 419-422.] Mondhatjuk tehát, hogy az elméleti megalapozottság fennáll, és elég régi ahhoz, hogy alaposan végiggondolt lehessen. Ennek ellenére komoly hiányosságot érzek ezen a területen, a valósághű (fogalmi szintű) modellezés szemszögéből. Az altípus csak egyetlenegy (eredeti) egyedhez kapcsolódhat. Ennek viszont megvan az a következménye, hogy ha egy jelenség több más jelenségnek lehet, vagy lehetne az altípusa, akkor azt ilymódon nem tudjuk valósághűen modellezni. Vegyük például a motorcsónakot: lehet az altípusa a vízijárműnek vagy a motoros járműnek, de *mindkettőnek* nem.

Érdekességképpen jegyzem meg, hogy az altípus fogalmának megfeleltethető „variálható rekord”-ot az elég régi Pascal programozási nyelv pl. ismeri annak ellenére, hogy nincs köze a relációs adatbáziskezeléshez, lévén általános (oktatási) célú programozási nyelv.

Inhomogén 1:1 kapcsolat

Az 1:1 kapcsolatok modellezése problémás terület. A problémát technikai síkon az okozza, hogy az 1:N-es kapcsolatok úgy valósulnak meg, hogy a kapcsolat N-es végén szerepeltetjük idegen kulcsként (kapcsolóként) az 1-es oldali azonosítót. Az 1:1-es kapcsolatnál ezt vagy szimmetrikusan kellene megtenni, ami redundáns, vagy pedig az egyik irányt kitüntetni a másikhoz képest, ami viszont egyensúlytalan állapotot eredményezhet, és nem valósághű.

Az ilyen kapcsolatban lévő egyedtípusok mechanikus összevonása modellelméletileg helytelen, hiszen különböző lényegekről van szó. Az 1:1 kapcsolat 1:N-es kapcsolattá alakítása (ahol N értéke éppen pont 1) a mellérendeltséget egyensúlytalanná teszi. A mellérendeltség megmarad, ha a két jelenség közé egy kapcsoló egyedtípust iktatunk be (mintha N:M felbontásról lenne szó), de ez sem tekinthető általános érvénynyel jó megoldásnak, és ráadásul a kezelés bonyolultabbá válik.

Ez viszont helytelen szemléletet takar, ugyanis minden további nélkül elképzelhetők olyan különböző (!) jelenségek, amelyek ilyen viszonyban vannak egymással, tehát - az adatmodellezés valósághűségének követelményét szem előtt tartva - két különböző egyedtípussal kell azokat modellezni, amelyek következésképpen 1:1 kapcsolatban vannak egymással. (Vegyük például a felnőtt személyek és pillanatnyi személyi igazolványaik példáját.)

A jelenlegi adatbáziskezelők elméleti korlátai

Egy mondás szerint az elmélet és a gyakorlat között nincs különbség - elméletileg legalábbis, mert a gyakorlatban azért akad. Sajnos ez a megállapítás igaz a relációs adatbáziskezelés elméletére és gyakorlatára mind a mai napig. Számos olyan eleme van a relációs adatbáziskezelésnek, amely annak elméletében régóta kidolgozott, pontosan leírt, mondhatni nyilvánvaló, és a gyakorlatban - a relációs elvű adatbáziskezelők (RDBMS) - máig nem, vagy nem megfelelően valósítják meg azokat.

Ez a jelenség a kezdetektől kíséri a relációs adatbáziskezelést. Az IBM eleinte nem volt hajlandó a relációs modellt alkalmazni, mert féltette az IMS/DB-ből jövő bevételét. Ahogy Codd a már idézett előszavában írja: tízéves küzdelmébe került a relációs elvű megközelítés elfogadtatása a szakmai és üzleti közvéleménnyel. A szoftvergyártók több esetben felcímkézték a „relációs" jelzővel termékeiket, de azoknak nem feltétlenül volt bármi közük a Codd-féle relációs elvhez, általában a meglévő régi eszközöket próbálták meg felruházni valamilyen mértékben a relációsra hasonlító tulajdonságokkal. Mivel a relációs adatbáziskezelést időben, illetve logikailag megelőzi a fájlkezelésen alapuló megközelítés (kidolgozott elmélet nélkül is), ez valamennyire érthető, magyarázható, ha nem is helyeselhető.

Természetesen (?) ez a hozzáállás a mai napig felismerhető helyenként. Elvégre ha egy szoftvergyártó temérdek munkát (pénzt) fektet be egy termékébe, annak forgalmazásában érdekelt, és nem abban, hogy azt részben vagy teljesen kidobja és esetleg teljesen új fejlesztésekbe fogjon. Továbbá abban is érdekelt, mégpedig erőteljesen, hogy a jövőben is legyen árbevétele, amit pedig a meglévő szoftvertermék egyre újabb és újabb változatának piacra dobásával tud biztosítani, aminek közvetlen következménye, hogy ezek egyike sem lesz teljes értékű és hibátlan. Mivel szoftvert - az anyagi javakkal ellentétben - nem lehet élettartamra méretezni, ezért egy megoldás marad: tartalékolni (esetenként megteremteni) a következő verzióban kijavítandó hibákat és megvalósítandó továbbfejlesztési ötleteket. Ez egyebek mellett avval is jár, hogy egyre több és több, változatos technikai elemmel bővül az eszköztár az elméleti teljesség és megalapozottság elérése helyett. Természetesen az üzleti szempontoknak ez a túlburjánzása nemcsak, sőt talán nem is elsősorban az adatbáziskezelőkre igaz, de problémánk szempontjából ez közömbös.

A relációs elvű kezelők korlátaiból, problémáiból mutatok be az alábbiakban néhányat.

Kényszerpálya

Vannak olyan hiányosságok és/vagy hibák, amelyeket rövidebb vagy hosszabb ideig fenn kell tartani a felhasználók érdekében, pontosabban a kompatibilitás érdekében. Ezt már a kezdet kezdetén Codd is említi: „Ráadásul az adatbáziskezelő termékek eddigi tervezési hibái is hátráltatják a fejlődést - gyakran szükséges ezen hibákat támogatni annak érdekében, hogy védjük a felhasználók alkalmazói programokba történt befektetéseit."[84] [F4296 p. viii.] Jellemző példaként legyen szabad említeni az üres értékek (NULL), illetve a 2-3-4 értékű logika problémakörét.

Egységes kezelés

Az adatbáziskezelés kialakulásának absztrakciós folyamatáról[85] fentebb leírtak szerint az „igazi" megoldás az, ha az adatbázisban kezelni szándékozott ismeretekre vonatkozó összes adatot (a meta-adatokat) egységesen, az adatbáziskezelő szoftverrel kezeltetjük, összhangban az ún. 100%-os elvvel[86]. Ez sajnos még ma sem általános, illetve nem teljes értékűen megvalósított, annak dacára, hogy már Codd a relációs modell fontos tulajdonságaként emeli ki ezt[87], mint fentebb már idéztem. Halassy Béla közlése (2001-ben): adatbáziskezelők és CASE-eszközök szakkiállításán *minden* kiállított termék esetében ki tudott mutatni elméleti hibát vagy hiányosságot."

Példának okáért a relációs adatbáziskezelés zászlóshajójának számító Oracle sem különbözteti meg mind a mai napig a hiányzó adatot[88] aszerint, hogy az ismeretlen, vagy pedig nem értelmezhető.

Értéktartomány

Értéktartománynak egy tulajdonságtípus *általánosan* felvehető értékeinek a halmazát nevezzük, függetlenül attól, hogy az adatbázisban pillanatnyilag található-e példa (előfordulás) minden lehetséges értékre, avagy sem. A reláció formális matematikai meghatározását véve: az S1, S2, ..., Sn - nem feltétlenül különböző - halmazokon R egy reláció,

84 „Additionally, errors made in the design of DBMS products along the way are also hindering progress - often it is necessary to continue to support those errors in order to protect a customer's heavy investment in application programs."
85 L. Ismeretkezelési módok, 40. old.
86 L. 25. old.
87 L. a Tulajdonságok és egyedek nevei c. részben, a 54. oldalon.
88 L. Üres értékek, 48. old.

ha ez egy halmaza azon rendezett n-eseknek, amelyek első eleme az S1, második eleme az S2, n-edik eleme az Sn halmaz eleme. Másképpen mondva az S1, S2, ..., Sn fölötti R reláció egy részhalmaza az S1xS2x...xSn Descartes-szorzatnak.

Az értéktartomány (domain) alapvető jelentőségű - lenne - a relációs adatbáziskezelésben. Ismét Coddot idézem, amikoris arról ír, hogy az értéktartományok jelentősége még az elsődleges és az idegen kulcsokénál is nagyobb: „A relációs modell számos tulajdonságának teljes támogatása az értéktartományok fogalmának teljeskörű támogatásától függ. Az értéktartományok teljes támogatásának néhány előnye következik. (...) az értéktartományok a ragasztó, amely egyben tartja az adatbázist. Jegyezzék meg: értéktartományról beszélek, nem elsődleges és idegen kulcsokról. A relációs modellben a kulcsok fogalma fontos további és szakosított ragasztót jelent."[89] [F4296 p. 45.] A relációs modell számos tulajdonságának teljesülése áll vagy bukik ugyanis az értéktartományok következetes alkalmazhatóságán.

Ebből következik, hogy nemcsak elvárható egy, a relációs jelzőre számot tartó adatbáziskezelőtől, hogy lehessen benne értéktartományokat definiálni, hanem megkövetelhető. Sőt, még azt is elő kellene tudni írnunk az értéktartományok vonatkozásában, miszerint - Halassy példáját idézve [F4260 p. 94.] - a munkanap (mint értéktartomány) a dátum értéktartomány részhalmaza.

A definitív (adatbázisszerű) kezelés lényege, amint azt a 100%-os elv[90] is kimondja, hogy *minden,* adatainkra vonatkozó ismeretet a (fogalmi) modellben kell rögzíteni. Ebből az is következik (következne), hogy a fogalmi modell *minden* tényezőjét ugyanazon adatbáziskezelőmotor kezeli, mint az alkalmazási adatbázist. Így lenne meg a lehetősége annak, hogy a munkanap mint értéktartomány a dátum részhalmazaként definiálható, anélkül, hogy procedurális módon kellene ezt megtenni - azaz fontos kelléke az adat-program függetlenségnek.

Szerepnév

A szerepnév nem más, mint az egyedtípuson belül sajátos értelmezésben használt általános tulajdonságtípus. Például ha egy lineáris vál-

89 „Full support for many of the features of the relational model depends on full support of the domain concept. Some of the advantages of supporting domains fully follow. (...) domains are the glue that holds a relational database together. Notice that I said domains, not primary keys and foreign keys. The concept of keys in the relational model provides an important additional and specialized kind of glue."

90 L. 25. old.

lalati hierarchiát szeretnénk leírni, akkor célszerű módon egy homogén visszamutató (1:N) viszonyt használnánk. A lineáris vállalati hierarchia esetén bármely dolgozónak pontosan egy közvetlen főnöke van (kivéve természetesen a legnagyobb főnököt, akinek egyáltalán nincsen főnöke - az adott cégnél legalábbis). Az ilyen szervezeti forma fa-gráffal írható le. Legyen egyedtípusunk neve DOLGOZÓ(dolgozóazonosító, dolgozónév, ..., *főnökazonosító*). Ebben a példában a „főnökazonosító" a dolgozóazonosító szerepneve. Erre két okból van szükség.

Egyrészt mert pusztán formálisan szemlélve a helyzetet nem engedhető meg az, hogy egy egyedítpusban két azonos nevű tulajdonságtípus legyen, márpedig a kapcsolatot a relációs modellben a fölérendelt egyedtípus azonosítójának az alárendeltben kapcsolóként való szerepeltetése teremti meg. Esetünkben, mivel a fölé- és az alárendelt egyedtípus egybeesik, a „dolgozóazonosító" kétszer szerepelne: elsődleges kulcsként (azonosító) és idegen kulcsként (hivatkozás). A névazonosság problémáját feloldja a szerepnév használata.

Másrészt pedig szükségünk van a szerepnévre a modellalkotás valósághűségének követelménye miatt is. A szerepnév itt kifejezi azt a sajátos viszonyt, hogy az adott (a „dolgozóazonosító" adott értékével azonosított) dolgozónak a(z adott főnökazonosító értékével) hivatkozott másik dolgozó nem akármilyen kollégája, hanem a *főnöke*.

A szerepnév előfeltétele számos további szolgáltatás meglétének, nélküle nincs pl. (teljes értékű) visszamutató kapcsolat vagy altípus-kezelés sem.

Homogén kapcsolatok

A kapcsolatok a legtöbb esetben inhomogén kapcsolatok, azaz különnemű egyedtípusok közöttiek. Vannak azonban olyan esetek, amikor nem kerülhetők meg a homogén kapcsolatok. Ezek egyik tipikus esete az előző pontban példaként említett 1:N homogén kapcsolat esete. A másik eset a homogén M:N kapcsolat esete. Azért kell ezeket külön említeni, mert itt a kapcsolat homogén mivoltából adódó külön érdekességei vannak a szerkezeti épségnek.

Az 1:N esetben figyelemmel kell(ene) lenni arra a sajátosságra, miszerint itt fa-gráfot írunk le: azaz körmentesnek kell lennie, vagyis nem fordulhat elő, hogy valakinek a beosztottja a főnökének a főnöke legyen, egyszerűbb esetben az, hogy valaki önmaga főnöke legyen.

A homogén N:M kapcsolatok tipikus alkalmazási területe a darabjegyzék vagy családfa jellegű helyzetek modellezése. A darabjegyzék lényege, hogy az elemi alkatrésztől (pl. kerékanya) a teljes összeszerelt

rendszerig (pl. menetkész gépkocsi) pontosan leírja, hogy a rendszer milyen részegységekből épül fel, azok milyen kisebb részekből állnak, egészen az eleminek tekintett alkatrészekig. Ez esetben is van a szerkezet jellegéből adódó különös korlát: a valóságban nem fordulhat elő az, hogy egy részegység önmagába épül be. Ez ugyanis végtelen ciklust jelentene, viszont inhomogén esetben efféle probléma nem léphet föl.

Relációs elvű adatbáziskezelőink nem mindegyike tudja ezeket a szerkezeti finomságokat kezelni. Ráadásul azoknál, amelyek a szerepnév fogalmát sem ismerik, a programozó jóindulatára van bízva, hogy pl. a kulcs és az idegen kulcs mezőket azonos típusúnak és méretűnek deklarálja-e, máris (ismét) megsértve a 100%-os elvet[91]. Ugyanis homogén viszonyok esetén a kapcsoló tulajdonságtípusok mindig szerepnevek.

Csoportok

A csoportok problémáját már említettem az előző részben.[92] Kezelőink, legalábbis azok egy része nem tud csoportokat kezelni, azaz ebben a vonatkozásban maradéktalanul megfelel a Codd-féle követelménynek. Holott legalább néha szükség lenne rá a modellezés valósághűségét, illetve az azonosíthatóság követelményét[93] szem előtt tartva. Ha ezt a lehetőséget az adott kezelő nem biztosítja, akkor az evvel kapcsolatos igényeket procedurálisan, esetleg beépített függvények alkalmazásával lehet kiszolgálni, ismét csak megsértve a 100%-os elvet[94].

Altípusok, 1:1 kapcsolatok

Bár az altípusok szükségességét már kezdetben világosan megfogalmazták, annak technikai megvalósítása nem általános. A kezelők egy része nem ismeri ezt a fogalmat, nem szolgálja ki az evvel kapcsolatos különleges többletigényt. Az altípus ugyanis 1:1 kapcsolatban van az eredeti egyedtípussal, amely lentről fölfelé kötelező (azaz az altípusbeli előfordulásnak mindig van fölérendeltje), fordítva viszont opcionális. Ez azonban nem „közönséges”, általános 1:1 kapcsolat, hanem különleges abban az értelemben, hogy öröklődés is van: az egyedaltípust nem csak a saját tulajdonságtípusai jellemzik, hanem a fölérendelt egyedtípus tulajdonságtípusai is. Csak érdekességképpen említem, hogy pl. a

91 L. 25. old.
92 Csoportok és atomi értékek, 57. old.
93 L. a Kulcs, idegen kulcs, kapcsoló c. részt, 64. old.
94 L. 25. old.

Pascal programozási nyelv - dacára általános célú programnyelv mivoltának - ezt a jelenséget a variálható rekord fogalmának alkalmazásával réges-régen megoldotta.

Az öröklődés az ún. objektum-orientált megközelítéseknek ugyan alap-sajátossága, viszont általános érvényű jelenségként nem biztos, hogy több problémát old meg, mint amennyit fölvet. Az igazi probléma az ún. objektum-orientált megközelítésben szerintem ott rejlik, hogy nem elegendő az eszköz kezelési módjának ilyennek lennie, az adott eszköz egész „alapfilozófiájának" kellene objektum-orientáltnak lennie, sőt, mi több: a valóság általa kezelt részének kellene olyannak lennie, amely jobban írható le az ún. objektum-orientált módon mint hagyományosan.

Nincs tudomásom azonban mindmáig arról, hogy ilyen alapelvű elméleti alapozás - a relációshoz hasonlóan - egyáltalán létezne. A fenti megfogalmazásban az „úgynevezett" kitétel azt próbálja érzékeltetni, hogy maga az elnevezés nem a legszerencsésebb, ugyanis a kifejezés szó szerinti értelmezését alapul véve minden (vagy legalábbis majdnem minden) megközelítési módra alkalmazható. A világ modellezendő részeit, vagy azok modelljét egyaránt nevezhetjük ugyanis objektumoknak - ha ennél kifejezőbb elnevezés nem jut eszünkbe. Hasonlóan az UoD-hez[95]. A jelenségek el-, illetve megnevezése fontosabb annál, hogysem ilyen könnyedén lehetne venni.[96]

Az altípustól független 1:1 kapcsolatok is problémásak abban a tekintetben, hogy vannak irányzatok, amelyek - tévesen - megkövetelik ezek összevonását.[97]

Üres értékek

Az üres értékek problémáját a relációs elmélet korlátai című részben bemutattam. A relációs kezelők ez esetben igazodnak az elmélethez, a NULL használata az egyetlen „gyári" lehetőség, vagy maradnak a procedurális megoldások.[98]

Kulcs, idegen kulcs, kapcsoló

A relációs elvű adatbáziskezelés legfontosabb alapeleme - érdekes módon - *nem* az elsődleges kulcs és az idegen kulcs, amelyek a kapcso-

95 „university of discourse", „az, amiről szó van". A szaknyelvben a magyar „izé" megfelelőjeként használatos.
96 L. a fogalomalkotásról szóló részt, 140. old.
97 L. az Inhomogén 1:1 kapcsolatok c. részt, 58. old.
98 L. 48. old.

latokat valósítják meg, hanem az értéktartomány[99]. Az értéktartományok teljes értékű megvalósításán múlik számos egyéb elvárás teljesülése vagy lehetetlensége. Codd írja: „Az értéktartományok elve igen fontos szerepet játszik a relációs modellben annak kezdetei óta. (...) *Az értéktartományok támogatásának elmulasztása napjaink relációs adatbáziskezelőinek legsúlyosabb hiányossága.*”[100] [F4296 p. 43.]

Evvel együtt is komoly hiányosság, hogy lehet táblát létrehozni elsődleges kulcs megadása nélkül, márpedig axiómaszerűen mondjuk ki, hogy *minden* egyedtípusnak kell legyen azonosítója. „RS-8, Az adatbáziskezelőnek meg kell követelnie, hogy minden egyes táblának legyen egyetlenegy elsődleges kulcsa. Ebben az egyszerű vagy összetett oszlopban minden értéknek különböznie kell a többitől minden időpillanatban. Az összetett oszlop egyetlen összetevőjének értéke sem lehet sosem ismeretlen.”[101] [F4296 p. 35.] „...azt mondja ki, hogy minden egyed-előfordulást meg kell tudni különböztetni egymástól.” [F4260 p. 74.]

Evvel ellentétben minden további nélkül lehet táblákat létrehozni azonosító (elsődleges kulcs) nélkül. Az idegen kulcsok szerepe, kezelése nem magától értetődő még ma sem, még akkor sem, ha a szóban forgó szoftvert adatbáziskezelőnek nevezik. Történik (van) mindez még ma, 2009-ben is, kerek negyven esztendővel Codd első, a relációs adatbáziskezelésről írott dolgozata után. Codd már ekkor (nagyon helyesen!) foglalkozik az adatbázis szerkezeti épségének problémájával. Későbbi, összefoglaló könyvében erre így emlékezik vissza: „Nagy hangsúlyt fektettem a kereskedelmi adatbázisok szerkezeti épségének megőrzésére, és így teszek ma is. Jelen könyvemben a 13. és a 14. fejezetet kizárólag ennek a tárgynak szentelem.”[102] [F4296 p. v.]

Számos olyan eset van, amikor összetett kulcsra van szükség, dacá-

99 Domain, l. 60. old.

100 „The concept of domains has played a very important role in the relational model since the model was conceived. It is not overstating the case to say that the domain concept is fundamental. It participates crucially in the definition of numerous features of RM/V1 and RM/V2. Consequently, many of these features cannot be fully supported by a DBMS unless that DBMS supports domains. Omission of support for domains is the most serious deficiency in today's relational DBMS products.”

101 „For each and every base R-table, the DBMS must require that one and only one primary key be declared. All of the values in this simple or composite column must be distinct from one another at all times. No value in any component column is allowed to be missing at any time.”

102 „I placed a great deal of emphasis then on the preservation of integrity in a commercial database, and I do so now. In this book, I devote Chapters 13 and 14 exclusively to that subject.”

ra annak, hogy elvileg nem, gyakorlatilag nem mindig van lehetőség a csoportok kezelésére. Előfordul az is, hogy adott kezelő feltételezi, hogy az összetett kulcs részei kapcsoló szerepűek, márpedig ez nem feltétlenül szükséges, tehát ez nem követelhető meg a kizárólagosság igényével.

Ismétlődő sorok

Az ismétlődő sorok problémája nem független a kulcsok problémájától. Ha ugyanis megkövetelné a kezelő az elsődleges kulcs kijelölését minden tábla esetében, akkor (a kulcs fogalmából következően) lehetetlen volna két teljesen egyforma sor előfordulása.

A relációs elmélet megalapozásakor Codd már kikötötte, hogy ismétlődő sorok nem fordulhatnak elő.[103] [F4296 pp. 32-33.] Ez nem téveszthető össze az ismétlődő adatok vagy adatcsoportok problémájával. Ez a matematikai alapozáson nyugszik, azaz a relációs modellbeli *reláció* a matematikai fogalomhoz hasonló abban az értelemben, hogy ha azt táblaként képzeljük el, az a következő tulajdonságokkal bír: a) minden sora egy rendezett n-es, b) a sorok sorrendje közömbös, c) minden sor különböző. [F4296 p. 2.]

Ennek ellenére nem feltétlenül kötelező az elsődleges kulcs kijelölése, és minden további nélkül lehetnek ismétlődő sorok egy táblában. Annak ellenére, hogy a matematika formális szabályait nem lenne szabad figyelmen kívül hagyni. Ezt 1. sorszám alatt köti ki Codd,[104] a korábbi adatbáziskezelő megvalósításokban föllelhető számos tökéletlenségre („numerous blunders”) tekintettel.

Chris Date ugyanezt az álláspontot képviseli. Database in Depth c., bő kétszáz oldalas munkájában külön fejezetet szentel a kérdésnek. [F4280 pp. 48-52.]

Szerepnevek

Vannak modellezési irányzatok, amelyek általánosan, minden esetben kötelezőként írják elő a szerepnevek használatát. Evvel nem lehet maradéktalanul egyetérteni, már csak a szerepnév fogalmának meghatározása okán sem.

Technikai adatok

Általában nem támogatják a kezelők az elsődleges és másodlagos adatok megkülönböztetését, értve a „másodlagos adat” kifejezés alatt

103 „RS-3 Duplicate Rows Prohibited in Every Relation.”
104 „RD-1 Non-violation of any Fundamental Law of Mathematics” [F4296 p. 351.]

az elsődleges adatok - amelyek kezelésére az adatbázist létrehozták - kezelése során, illetve kapcsán keletkező, technikai adatokat (pl. ki, mikor módosította az adott elsődleges adatot).

Adatbázisgépek

Az adatbáziskezelők és az operációs rendszer egyes tevékenységeket egymással részben vagy egészben párhuzamosan végeznek (pl. a háttértár-kezelés egyes tevékenységeit). Ezért jogosan merül föl az a kérdés, hogy nem volna-e célszerű az operációs rendszer és az adatbáziskezelő egybeolvasztása a nagyobb hatékonyság érdekében. Az ilyen céltechnika az adatbázisgép.

Az SQL-nek Codd által felrótt hiányosságai

Van az SQL-nek néhány olyan hiányossága, amelyeket már Codd is felrótt, és mind a mai napig nem születtek rá megnyugtató megoldások, dacárra annak, hogy komoly, elméleti súlyú problémákról („serious flaws") van szó. Codd a teljes 23. fejezetet szenteli ennek a problémakörnek már többször idézett könyvében. [F4296 pp. 371-389.] Ezek közül az ismétlődő sorok problémáját már bemutattam fentebb.[105] A harmadik ilyen probléma a háromértékű, még inkább a négyértékű logika támogatásának hiánya, az üres értékek problémájától[106] el nem választható módon, hiszen pont az üres értékek problémája világít rá ennek szükségességére. A Codd által másodikként említett probléma a pszichológiai és a logikai szint elkülönítésének hiánya.[107] Ez utóbbi az egymásba ágyazott lekérdezések és a „közönséges" select (equi-join) eredményeinek különbözőségét érinti - ismétlődő sorok esetén.

Előtérben a technika, háttérben az elmélet, a modellalkotás és a szabványok

Napjainkban még mindig (vagy egyre inkább) a technika szerepe látszik elsődlegesnek, a jó modellezési szemlélet, gondos tervezés, általában a megfontolt gondolkodás, sőt annak támogatása is a háttérbe szorul. (Vagy sosem is volt előtérben?) Egy hasonlattal élve: egy mai rákoskeresztúri sírkövesnek olyan technika áll a rendelkezésére, amelyről Michelangelo nem is álmodhatott annak idején. Ennek ellenére úgy tű-

105 L. 56. old.
106 L. 48. old.
107 „they [i.e. all present versions of SQL] fail to separate psychological features from logical features"

nik, a „termék” minőségét mintha nem elsősorban a rendelkezésre álló technika határozná meg elsősorban, hanem az azt kezelő ember. Az összehasonlítás indokolt: hiába a legjobb adatbáziskezelő, hiába a nagy teljesítményű gépek, ha az adatmodell rossz, mert az *alkotó ember* nem állt helyzete magaslatán.

Mivel az információs rendszerek, illetve az azok alapjául szolgáló adatbázis minősége áll vagy bukik az adatmodellezés jóságán (más, nem elhanyagolható tényezők mellett) és mert a jelenleg meglévő eszközök is tökéletlenek, az ember szerepe felértékelődik. A technika hiányosságait a (szak)embereknek kellene pótolniuk. „A rendszerek és az illesztőegységek elkészítése, működtetése és karbantartása gyakran drágábbak mint kellene. Az is előfordul, hogy inkább korlátozzák az üzletmenetet mintsem segítenék azt. Ennek egyik fő oka, hogy az alapul szolgáló adatmodellek minősége szegényes.” [F4319 p. 7.]

„...fényévekkel vannak elmaradva a jelenlegi eszközök egy ideálisnak nevezhetőtől: ráadásul indokolatlanul, hiszen nem O(N)-ben kell prímet találni vagy valami hasonló problémát leküzdeni hozzá. (...) Még az sem megoldás, ha minden jelenlegi eszköz spéci okossága egybegyúrható lenne. Valahol az ipar is felelős, hogy ilyen szintű gányolás tapasztalható a világban...” [F4304]

Történeti kitekintés

A '70-es években, a kezdetekkor az adatbázistervezés, illetve az adatmodellezés igen fontos volt. Egyrészt azért, mert az akkor rendelkezésre álló technika igencsak szűk keresztmetszet volt: mágnesszalagos háttértárral nem igazán lehet adatbáziskezelést csinálni, relációsat meg végképp nem.[108] Az IBM 1973 márciusában jelentette be az IBM 3340 lemezegységet, amely forradalmi újításokat tartalmazott (író-olvasó fej távolsága az adattároló felülettől másfél nanométerre csökken, emiatt muszáj légmentesen lezárt térben üzemelnie, nem érintkezhet a gépterem levegőjével, mint a korábbi, cserélhető lemezegységek). Egy kisebb és egy nagyobb kapacitású változatát gyártották, ezek tárolókapacitása 35, illetve 70 MB volt, 885 KB/sec adatátviteli sebességgel és 25 msec átlagos elérési idővel. [F4328] Azaz mondhatjuk, hogy az elmélet - ekkor még - megelőzi a technikát. Az IBM korábbi, 3330-as lemezegységét 1970 nyarán jelentették be. A korábbi lemezegységek, mint pl. az IBM 2311 7,25 MB tárolókapacitású volt lemezcsomagonként. [F4330 pp. 29-31.]

108 Vö. 43. old.

Tárgyi feltételek híján, vagy azok nagyon szegényes volta esetén mindenképpen az elmélet lesz az elsődleges - nincs más lehetőség. Ugyanezért lesz (volt) igen fontos a minél gondosabb tervezés, hiszen az erőforrások igen szűkös volta a legkisebb pazarlást sem tette lehetővé. Nem véletlen tehát, hogy Codd ennyire megalapozott, ennyire részletekbe menő módon dolgozta ki az adatbáziskezelés relációs elméletét, és ugyancsak nem véletlen, hogy az adatmodellezés területén is igen korán elvégezték a máig ható alapozást - Chen már említett tanulmánya az egyed-kapcsolat (E/R) modellről 1976 márciusában jelent meg. Míg azonban az adatbáziskezelők működési elve - relációs alapokon - kidolgozható, sőt kidolgozandó tisztán elméleti alapokon, matematikai és logikai megfontolások alapján, addig a hétköznapi értelemben vett és a hétköznapi igényeknek megfelelő adatmodellezés már sokkal kevésbé. Visszatérve Michelangelo példájához: bármilyen szerszámot a kalapácstól a sarokköszörűig meg lehet és kell tervezni, hogy az gyártható legyen. Viszont általános eszközt és módszert adni arra, hogy egy tetszőleges kőtömbből hogyan lehet tetszőleges szobrot csinálni, már jóval nehezebb - de inkább lehetetlen[109].

A '80-as években történtek kísérletek Chen egyed-kapcsolat modelljének a kiterjesztésére. Közben a hardver fejlődése jelentősen meggyorsult. 1980-ban jelentették be az IBM 3380 közvetlen elérésű háttértárat, amely jelentős teljesítménynövekedést mutatott a korábbiakhoz képest. Tárolókapacitása négyszerese a korábbiaknak (2,5 GB), átviteli sebessége pedig duplája a korábbinak (3 MB/sec). [F4332] 1981-ben jelent meg az IBM PC [F4331] és evvel már a személyi számítógépek sikertörténete kezdődik. A hardver fejlődése ugrásszerű, egyre kevésbé jelent korlátot. Az általános szoftverfejlődésben eljutunk az objektum-orientált programnyelvek megjelenéséig.

A rákövetkező évtizedben az adatbázistervezésben egyre inkább megjelennek az ipari szempontok, az adatbázisszerkezet transzformációja, a minőség mérése. Ekkortól számíthatjuk az objektum-orientált modellezési eljárások megjelenését is. Az évtized elején több tucat modell-leíró nyelv létezik. A Booch, Jacobson és Rumbaugh által kidolgozott UML[110] 1997-ben válik az objektum-orientált modellezés szabványos eszközévé.

1976-ban az IBM-nél a SEQUEL Codd, Chen és Date munkásságának eredményeként jelenik meg. Ez lesz később az SQL (Structured Query

109 Dacára a Michelangelonak tulajdonított mondásnak, miszerint minden kőben benne van a szobor, csak a fölösleget kell lehántani róla.

110 Unified Modeling Language

Language). [F4272] Szabványosítási folyamatának főbb állomásai: ANSI (1986), ISO (1987) csak az alaputasításokra vonatkozóan. Az előfordítók és dinamikus SQL utasítások szabványosítása: ISO 9075:1989. Az ISO 9075:1992 (SQL2) már a bővített adattípusokat, értékszabályokat, kulcsdefiniálásokat stb. írja le. Az SQL3 1998/99-ben a rekurzív és objektumorientált adatszerkezetek mellett további relációs műveleteket épít be, mint pl. az outer join, natural join, difference, intersection, megjelenik a catalog és schema koncepció. További adattípusokat határoz meg (dátum típus[111] - igen, csak az SQL3-ban!). [F4335 pp. 173-175.]

Az ODMG[112] szabványok 1993-2003 között az objektumorientált adatbázisokra vonatkozó követelményeket határozzák meg.

Szerintem botrányos, hogy a tisztán halmazelméleti, ill. relációs algebrai alapokon nyugvó relációs adatbáziskezelés negyven év alatt a mai napig nincs elméletileg teljes értékű módon szabványosítva, és a nincs elméletileg teljes értékű relációs kezelő. A megvalósított, beépített elemek formális követelményei (szintaktika) a legtöbbször nem egységesek. Ugyanakkor a különféle relációs kezelők esetenként igen nagyszámú, a relációs kezelés szempontjából nem létfontosságú, technikai (kényelmi) szolgáltatással felvértezettek. Ezen tarthatatlan helyzetre egyetlen logikus magyarázat kínálkozik: az üzleti szempontok, ezen belül is a profitmaximalizálás túlzott térnyerése. Ennek alaposabb vizsgálata azonban nem tárgya a jelen dolgozatnak.

Kezelők fejlődése és sajátosságai

A kezdet kezdetén Codd leszögezte, hogy a relációs elvű adatbáziskezelő az adattárolás és visszakeresés fizikai szintjét teljes egészében el kell rejtse a felhasználó elől.[113] Ez nagyon helyes alapelv, szerintem az egyik oka a relációs adatbáziskezelés sikerének. Azonban van ennek is hátránya, mégpedig több okból. Ezen okok közül a legfontosabb az, hogy a logikai és a fizikai szint nehezebben választható szét, pontosabban sokkal inkább hatással vannak egymásra, mint mondjuk a fogalmi és a logikai szintek.[114]

A mai adadtbáziskezelők a fizikai szintet majdnem teljesen elrejtik a felhasználó elől. Felhasználó alatt most nem a végső felhasználót, az

111 L. a Csoportok és atomi értékek c. részt, 57. old.
112 Object Database Management Group
113 Az indexelés fizikai szintű tevékenység a három szintű megközelítésben, és nincs teljes összhangban Codd értelmezésével.
114 L. a Leképezések c. részben, 29. old.

adatbázisra épülő alkalmazás kezelőjét, hanem az adatbázis létrehozóját, illetve kezelőporgramjainak fejlesztőjét értve. Az adatbáziskezelők fizikai szintű megoldásainak dokumentáltsága általában nem kielégítő, azok nehezen hozzáférhetők, és úgy tűnik, mintha erre a területre már nem igazán jutna kellő figyelem. A dokumentációk általában nem elegendőek ahhoz, hogy adott konkrét helyzetben kísérletezés nélkül el lehessen dönteni, hogy melyik a gyorsabb és/vagy energiatakarékosabb megoldás a lehetséges megoldások közül.

Ugyancsak már Codd foglalkozik a lekérdezések adatbáziskezelők általi optimalizálásával is. Az egyes adatbáziskezelők ilyen irányú lehetőségei illetve korlátai talán még elmélyültebb ismeretét követelnék meg azoknak, mint a fizikai szintű műveletek ismerete. „...nem azt jelenti, hogy akár 2 tábla esetén egyértelmű, hogy milyen irányba[n] (érdemes) lekérdezni belőle (szűrés függő, INNER JOIN is szűrés). Oracle elég jól megcsinálta, hogy te csak felsorolod a fromba a táblákat, ő majd (...) kitalálja a sorrendet és irányt (láttam már Oracle oktatót rácsodálkozni, hogy LÉTEZIK JOIN?)” [F4305]

Megfigyelhető tehát egyrészt egy olyan fejlődési folyamat, amely a növekedő igények (mennyiség és teljesítmény vonatkozásában) kiszolgálását célozza, ugyanakkor, mintegy ezt ellensúlyozandó, olykor a legalapvetőbb elemek is hiányoznak vagy elégtelenek, mint pl. a kapcsolatok kezelése (elsődleges kulcs és idegen kulcs kapcsolat), különféle korlátok érvényesítésének lehetősége stb. Van olyan - közismert és jelentős piaci részesedésű - adatbáziskezelő, amelynek az alapértelmezett kezelőmotorja nem tudja lekezelni az idegen kulcs jelentette szerkezeti korlátokat.[115] Ugyancsak nem magától értetődő a homogén 1:N kapcsolatok (fagráf), a homogén M:N kapcsolatok (darabjegyzék) megfelelő kezelése. A fa (1:N), illetve a háló (M:N) bejárása pedig nemcsak a kezelőkön, de magán az SQL-en is kifog, pedig mindkét eset alapvető fontosságú adatmodellezési, adatszerkezeti elem.

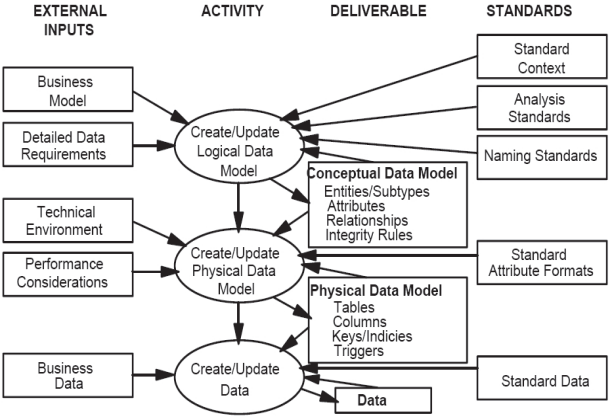
Említendő még - az emberi ismerethiányon túl - az, hogy nincs igazán kiforrott és minden esetre általánosan - és jól - alkalmazható fejlesztési módszer. Úgy vélem, hogy ilyent elég nehéz lenne elképzelni is, hiszen az adatbázistervezési feladatok és igények (is) olyannyira szerteágazók és sokfélék lehetnek, hogy azok teljes körét egyetlen, általános fejlesztési módszerrel eredményesen lefedni nem igazán lehetséges, vagy legalábbis gazdaságosan nem. Ez azonban puszta sejtés, a probléma végiggondolása - érdemes lenne pedig - meghaladja a jelen

115 L. Kulcs, idegen kulcs, kapcsoló, 64. old.

dolgozat kereteit. A jelen dolgozat alapgondolata pusztán annyi, hogy az SSADM használhatóságát, eredményességét nagymértékben megnövelné, ha abban megjelenne a három síkon - fogalmi, logikai és fizikai - történő modellezés illetve tervezés követelménye.[116]

Szabványok

A szabványok létét sokan hajlamosak az önálló alkotó személyiség többé-kevésbé értelmetlen és fölösleges korlátaiként fölfogni. Nem állíthatom azt, hogy ezen fölfogásban esetleg nincs valamelyes igazság. Azonban a szabványok léte és szerepe messze túlmutat ezen a problémán, és ha az „igazi programozó”[117] szemléleti síkja fölé tudunk egy kicsit emelkedni, akkor belátható, hogy az együttműködés, az együtt-dolgozás, a közös (informatikai) lét alapjairól van szó, amely minden résztvevő *közös* érdekét szolgálja, a közlekedés szabályaihoz hasonlóan. Még akkor is, ha adott kivételes (!) esetben az adott szabály, vagy annak alkalmazása esetleg valóban értelmetlen lenne. A szabványok egyszerűen a közös nyelvet hivatottak megteremteni a fejlesztés és a felhasználás résztvevői között.



5. ábra. Szükséges szabványok [F4319 p. 13.]

116 L. a Háromszintű megközelítés c. részt, 26. old.

117 L. pl. http://szabilinux.hu/orlando_unix/igazi.html - az internetes irodalom és humor egyik gyöngyszeme.

Nagyon fontosnak tartom azt leszögezni, hogy a megfelelő szabványok megléte ugyan nagymértékben elősegítheti a jobb minőségű és gazdaságosabb fejlesztési munkát, de annak - önmagában - nem szükséges és nem elégséges feltétele. Az, hogy önmagában nem elégséges feltétel, nyilvánvaló. De miért nem szükséges? Azért, mert a megfelelő modellezési, fejlesztési (üzemeltetési) szabványok megléte csak lehetőséget biztosít. Minden szabvány, szabály ugyanis pontosan annyit ér, amennyire azt *jól alkalmazzák,* ahogyan a szabályokat, szabványokat tartalommal megtöltik. Ezen a ponton viszont vissza kell csatolnunk az adatmodellezéshez, mint alkotó szellemi tevékenységhez: semmilyen szabály és szabvány nem helyettesítheti és nem pótolhatja a megfelelő modellezési szemléletet, illetve készséget. Ugyanakkor azonban hatékonyan elősegítheti és támogathatja a modellezés folyamatát - is.

Az SSADM módszertani problémái

A hagyományos életciklus-szemlélet szerint egy (az) információs rendszer a megvalósíthatósági tanulmánnyal kezdődik, majd a feltáró elemzés (analízis[118]), tervezés, kivitelezés, tesztelés, beüzemelés, használat lépései következnek, majd a hurok bezárul, a használat közben fölmerülő problémák, illetve újabb elvárások kiváltják a karbantartás szükségét, és kezdődik a ciklus újból a feltáró elemzéssel.

Az információs rendszerek életciklusát lehet ennél árnyaltabban, összetettebb módon is vizsgálni, azonban a végső váz mindig ez marad: tervezés - készítés - használat, a használat során fölmerülő igények kielégítése a meglévő rendszer továbbfejlesztésével vagy átalakításával - mindaddig, amíg nem muszáj vagy nem érdemes teljesen új alapokon teljesen új fejlesztésbe fogni.

A rendszer kifejlesztésére irányuló tevékenységeknek tehát valamilyen módon és mértékben célszerű ezt a folyamatot követniük. A fejlesztési folyamat ún. vízesésmodellje pont ezt a megközelítést követi. A modell klasszikus változata szigorúan soros végrehajtást ír elő, a következő munkafázis csak akkor kezdhető meg, ha az előzőt lezárták, és a megrendelő annak eredményét elfogadta. [F4282 p. 312.]

A fejlesztés klasszikus vízesésmodelljének a hátrányait hivatott ellensúlyozni a visszacsatolásos vízesésmodell, amely lehetőséget biztosít a visszacsatolásra, összhangban a „lépésenkénti finomítás” és a „vissza az ősökhöz” ősrégi, a programozásban is ismert és alkalmazott

118 L. Elemzés és analízis, 79. old.

alapelveknek megfelelően, vagy ha az ellenőrzési funkció valamely ponton hibát jelez.

Az első probléma - amely nem SSADM-specifikus - az, hogy két különböző területről van szó. Beszélünk egyfelől adatmodellezésről, logikai és fizikai tervezésről - fejlesztésről mint szűk értelemben vett informatikai munkáról -, emellett azonban nem szabad elfelejtkezni arról sem, hogy egy információs rendszer megtervezése, kialakítása, beüzemelése és működtetése, ez utóbbiba beleértve a karbantartását is, projektvezetési feladat is. Nyilvánvaló, hogy a sikerhez mindkettőre szükség van, de egyik a másik nélkül nem tudja a kívánt eredményt biztosítani. Nem szabad viszont összekeverni a két területet, és nem szabad, nem érdemes elvárni egy(etlen) eszköztől és módszertől, hogy ötvözze a projektvezetés és - mondjuk - az adatmodellezés feladatait.

Egy információs rendszer elkészítése, határidőre történő beüzemelése egyértelműen projektfeladat, mint ilyen, igényli a projekttervezés és -vezetés teljes eszköztárát, ugyanakkor azonban - nyilván - nem lehet figyelmen kívül hagyni a feladat sajátos - információtechnológiai - jellegéből adódó követelményeket és korlátokat, azaz nem lehet egy kalap alá venni - mondjuk - a hídépítéssel, és viszont.

Fejlesztési módszerek, módszertan

Ismét pontosítani kell a szóhasználatot. A „**módszer** fn Tudományban ált. érvényű tételekhez v. vmely eredményhez elvezető tervszerű eljárás.” [F4313 p. 964.] Raffai Mária a következő, összegző meghatározást adja a módszer fogalmára: „A módszer bizonyos feladatok elvégzéséhez szükséges, meghatározott feltételek között érvényes szisztematikus végrehajtási mód és ennek előírása.” [F4336 p. 284.] A *módszertan* kifejezés viszont, szó szerint értelmezve a „-tan” utótagot, nem más mint a „módszerek tana”, azaz a módszereket, esetünkben a szoftver-, illetve rendszerfejlesztési módszereket tudományos alapon és eszközökkel vizsgáló tudományág. A hétköznapi szóhasználatban azonban - mint annyi más helyen - itt is összekeveredik a kettő, a köznyelvi használat nem különbözteti meg határozottan és egyértelműen a két kifejezést ill. jelentéstartalmat, igen gyakran használják a „módszertan” szót „módszer” értelemben.

A számítógépek gyártásában az 1960-as évek (évtized) volt a forduló„pont”. Az elektroncsöves számítógépek ugyanis még a nagyon kevés kiválasztott eszköze csupán, nem is elsősorban az ipari értelemben vett felhasználás a cél, hanem magának a számítógépnek mint eszköznek, hardvernek a ki- és továbbfejlesztése. 1958-ban készül az első számító-

gép elektroncsövek helyett tranzisztorokból. 1960-ban születik meg a stratégiai döntés a majdani számítógépes hálózat felépítéséről (nem lehet kitüntetett szerepű, központi gép, a hálózat bármely részének megsemmisülése esetén a többi rész továbbra is működőképes), az első hálózat létrejötte pedig az évtized vége. 1969-70-ben négy számítógép alkotja az első hálózatot. Ekkoriban a szoftverfejlesztés még nem kerül előtérbe, az elsődleges cél a hardver működtetése, a legalapvetőbb feladatok megoldása, egyáltalán: operációs rendszer készítése. Ebben az időszakban a szoftverfejlesztéseket többnyire a hardvergyártók maguk végzik.

Az 1970-es években szoftverfejlesztés már jóval hangsúlyosabbá válik, megjelennek az első szoftverfejlesztő vállalatok is. Az egyre inkább ipari körülmények közötti, ipari követelményeknek megfelelő fejlesztés értelemszerűen magával hozta a fejlesztési megoldások, módszerek általánosítását, illetve ennek szükségszerűségét.

A cél ugyanis világos és egyértelmű: olyan eljárásra van szükség, amely a konkrét, pillanatnyi feladattól független, általános érvényű elemzési, értékelési, tervezési eljárás, amely a pillanatnyi feladatokra alkalmazva ha nem is garantálja a végtermék kifogástalan minőségét, de ezen cél elérésének valószínűségét lényeges mértékben megnöveli.

Ezt nagymértékben elősegítették a nagy kormányzati megrendelések is, amelyeknél egyre inkább megkövetelték a szabványos fejlesztési módszerek alkalmazását, így aztán azok iparági szabvánnyá is váltak. „Fejlesztési elv, paradigma alatt olyan, általánosan érvényes, a munkavégzést, munkastílust meghatározó hozzáállást, gondolkodásmódot értünk, amely lehetővé teszi az objektív valóság sajátosságainak és törvényszerűségeinek az általánosítását.” [F4336 p. 287.]

Valamely fejlesztési módszer egyben modell is, az általános fejlesztési eljárás modellje, és mint ilyenre igazak a modellről, modellezésről elmondottak is.[119] Ennélfogva nem szabad figyelmen kívül hagyni azt a sokszor háttérbe szorulni látszó körülményt, hogy minden modellnek, tehát adott fejlesztési módszernek is megvan a maga érvényességi köre, amelyen kívül nem lehet, ennélfogva nem is szabadna (fel)használni.

Informatikusként örülünk annak, hogy mára a fejlesztési módszerek és segédeszközök igen szép számmal vannak jelen környezetünkben (Raffai Mária 35-öt szed csokorba „CASE eszközök osztályozása” fejezetcím alatt [F4282 pp. 346-347.]), de óvakodjunk attól, hogy azokat,

119 L. 14. old.

vagy azok közül bármelyiket tévedhetetlen „csodafegyvernek” tartsuk, amely bármilyen körülmények között elvezet a kifogástalan minőségű végtermékhez. Ha így lenne, nem találkoznánk nap mint nap annyi elrettentő példával.[120] Az eszköz, a technika *lehet* szükséges feltétele a kívánt eredmény elérésének, de önmagában ritkán elégséges. A mögötte álló, azt értelmesen használni tudó, gondolkodó, problémamegoldó ember viszont nélkülözhetetlen, *mindig szükséges* feltétel.[121]

Személy szerint nem tudok hinni az „univerzális csodafegyverként” hirdetett eszközökben - ha azok esetenként mégoly hasznosak is -, nem tudok abban hinni, hogy egyetlen „integrált” eszközzel eredményesen és hatékonyan, általános érvénnyel át lehetne fogni a folyamatot (bármely) feladat megfogalmazásától az elkészült rendszer üzemeltetésével bezárólag. Mert minél általánosabb érvényű egy modell, szükségképpen annál sekélyebb is: minél nagyobbat fog át vízszintesen, annál kevesebbet függőlegesen és megfordítva. Természetesen nem azt akarom evvel mondani, hogy ezen módszerek, az azokat segítő eszközök fölöslegesek. Éppen ellenkezőleg, nagy szükség van rájuk, és nélkülük sokkal nehezebb dolgunk lenne. Pusztán arra szeretnék rámutatni ismételten, hogy nem a technika teszi az embert, hanem pont megfordítva, amint azt Halassy dr. is részletesen kifejti. [4314 pp. 25-38.] Ezt az álláspontot erősíti meg egy konkrét, az SSADM alkalmazásának tapasztalatairól szóló vizsgálat[122] is.

CASE-eszközök

A CASE betűszó, a számítógéppel támogatott rendszertervezés[123] rövidítése. A CASE-eszközök olyan, tervezést segítő, támogató programok, amelyek használatával könnyebben, gyorsabban, kevesebb hibával, egyszóval hatékonyabban lehet a tervezőmunkát végezni. Jó esetben. A CASE-eszközök el nem hanyagolható hányada - bár számosan tartanak igényt az elnevezésre - alapvető fontosságú szolgáltatásokat egyáltalán nem, vagy hibásan nyújt.

„Azokat a szoftverrendszereket, amelyek támogatják, illetve automatizálják a problémadefiniálási és -elemzési, a tervezési, a modellezési és a kivitelezési tevékenységet, amelyek elvégzik a tesztelési, a rendszerkövetési, a karbantartási és a minőségbiztosítási feladatokat,

120 L. a 19. oldalon az „Egyik eset (2007-ből)...” kezdetű bekezdéstől.
121 L. Michelangelo példáját az Előtérben a technika... c. rész elején, 67. old.
122 L. Az SSADM rövid története c. részben az „1999-ben vizsgálták...” kezdetű bekezdést.
123 Computer Aided Systems Engineering

amelyek dokumentálják a fejlesztést, valamint irányítják, ellenőrzik és összehangolják a fejlesztő projekt munkáját, számítógéppel támogatott fejlesztési eszközöknek, CASE rendszereknek nevezzük." [F4282 p. 344.] „I-CASE integrált fejlesztőkörnyezetnek azt a módszertani, szoftver- és eszközrendszert tekintjük, amely az alkalmazásfejlesztés teljes életciklusában automatizálja és koordinálja a munkafolyamatok egyes tevékenységeit." [F4282 p. 351.]

Raffai Mária fenti meghatározásában fontos elem a „módszertani, szoftver- és eszközrendszer" kitétel. Ugyanis problémánk több síkon szemlélhető és szemlélendő, legegyszerűbben közelítve is szó van egyrészt egy projekttervezési és -vezetési feladatról (határidők és költségkeretek!), illetve egy szoftvertervezési és -fejlesztési feladatról. A kettő nem cserélhető fel, nem keverhető össze, és egyik a másik nélkül nem kezelhető. A szoftverfejlesztési feladat ugyanis nem önmagáért való, öncélú és önmagában létező dolog, hanem határidőre, adott költségkeretből elvégzendő mérnöki munka. A fogalmi szintű modellezés, a logikai és fizikai szintű tervezés, majd a szoftver fejlesztése (tesztelése, javítása stb.) egészen más jellegű feladat, támogatása egész más jellegű eszközöket igényel, mint a projektvezetés alapvetően hálótervezési és annak figyelemmel kísérési feladata.

A szoftvertervezési és -fejlesztési folyamat is legalább olyan sokféle lehet, mint az általános mérnöki munka. Nem valószínű, sőt megkockáztatom: nem is lehetséges, hogy egyetlen általános célú eszközzel és módszerrel hatékonyan ki lehessen azt szolgálni. Épp ezért mindig az adott helyzetnek leginkább megfelelő eszközöket és módszereket kell kiválasztani. Ezek természetesen lehetnek, sőt legyenek is megfelelően dokumentáltak, szabványosak, a szükséges módon és mértékben testreszabottak, illetve testreszabhatók.

Raffai már idézett munkája szerint a magas szintű fejlesztőeszközök feladatai a következők: 1. Környezeti feltételek specifikálása, kezelése, dialógustervezés, képernyőszerkesztés és kommunikáció; 2. Adatspecifikációs, -lekérdezési és -manipulációs lehetőségek felkínálása; 3. Programtervezési támogatás; 4. Kódgenerálás; 5. Grafikus felhasználói felület, képernyőtervezés; Outputterv-generálás, jelentéskészítés. [F4282 p. 341.]

Érdekes megfigyelés, hogy a felsorolt elvárások között egyáltalán nem szerepel az adatmodellezés, illetve annak támogatása. Sajnos általánosnak mondható, hogy ez a terület nem kapja meg a jelentőségének megfelelő hangsúlyt.

A magas szinten automatizált, 4GL fejlesztések eredményeképpen a

programok lassúbbak, összetett, bonyolult rendszerek esetén az egész rendszer áttekinthetetlenné válhat, nagyobb erőforrást igényelnek. Összegezve: a fejlesztés lesz takarékosabb, az üzemeltetés pedig drágább. [F4282 p. 342.] Márpedig - jó esetben - egy információs rendszer életciklusában a használat a jóval jelentősebb szakasz, mint a fejlesztés, ennélfogva elvileg helytelen a fejlesztésen úgy takarékoskodni, hogy annak árát az üzemeltetés során kell hosszú időn keresztül, uzsorakamattal növelten megfizetni.

A CASE-eszközöket csoportosítani, osztályozni lehet különféle szempontok szerint. Ilyen szempontok pl. a képességek (milyen módszertani megoldásokat támogat), a bonyolultság (egyszerű szerkesztőtől integrált fejlesztőrendszerig), a fejlesztési fázisok támogatása (mely fázisokat támogatja, milyen módszertani megközelítéssel). A CASE-eszközök fontosabb összetevői: Üzleti rendszerek modellezése, elemzés-tervezés, implementáció (programgenerálás), tesztelés, karbantartás, projektmenedzsment, repository. [F4282 pp. 344-345, 352-359.] Sajnos az adatmodellezés támogatását annak fontosságához illő módon itt sem találjuk meg.

Az adatmodellezést támogató CASE-eszközökkel szembeni legfontosabb elvárások az alábbiak: grafikus ábrázolás, ezen alapuló tervezés; egyidejű dokumentálás; a formális alapon kimutatható vagy vélelmezhető hibák megkeresése; legalább a lehetséges tartalmi elemzések elvégzése, a szemantikus normalizálás[124] támogatása; a fogalmi modellezés és az azon alapuló logikai és fizikai szintű tervezés támogatása; tervváltozatok párhuzamos kezelése; többféle jelölésrendszer, szabvány, konvenció választhatósága, beleértve saját szabályrendszer felállíthatóságát; a relációs adatbáziskezelés teljes elméleti eszköztárának támogatása, függetlenül attól, hogy mely adatbáziskezelő ebből mennyit tud megvalósítani; elsődleges és másodlagos (technikai) adatok megkülönböztetése; metaadatbázis egységes rendszerben történő felépítése és kezelése; a fizikai terv alapján az üres adatbázis generálása kiválasztott adatbáziskezelő eszközhöz igazítva; többféle adatbáziskezelő adatbázisából, ill. az azt létrehozó SQL-parancsfájlokból a grafikus adatbázisterv létrehozása (reverse engineering); a csapatmunka-támogatása; változatok kezelése stb.

124 L. a „Halassy Béla szemantikus normalizálásról ír...” kezdetű bekezdést, 38. old.

Elemzés és analízis

Ezen a ponton ismét szükséges a fogalmakat, illetve a szóhasználatot pontosítani. A köznyelvben ugyanis az analízis és az elemzés rokonértelmű kifejezéseknek számítanak, jobb esetben az elemzést az analízis mint idegen szó magyar megfelelőjének tartják. Azonban világos különbséget kell tenni két, különböző jelentéstartalom között. Az egyik az a jelentéstartalom, amely a „részekre bontás, alkotóelemekre szétszedés" értelmű. A másik jelentéstartalom pedig „visszafelé oldás, fordított irányú megoldás, vagy inkább fordított irányú munka". [F4338 p. 186.] Az utóbbi jelentéstartalom magában foglalja, illetve feltételezi az intuíciót, a tervezést, a szintézist, azaz nem a szintézis ellentéte. Alkotó, teremtő tevékenység, hasonlóképpen a modellalkotáshoz[125], amellyel szoros kapcsolatban van. A modellalkotás általában, ezen belül az adatmodellezés specifikusan ugyanis feltételezi és igényli a modellezett valóság ilyen jellegű, alkotó módon történő elemzését, analizálását.

Az SSADM rövid története

1980-81-ben az Egyesült Királyság Kormányának Központi és Számítástechnikai és Távközlési Ügynöksége, a CCTA[126] megbízásából az LBMS[127] cég dolgozta ki az első változatot saját korábbi tervezési módszerének (LBDB módszer, 1977.) alapján. A CCTA követelményei egyszerűek és világosak voltak: a fejlesztési módszer legyen önellenőrző, kipróbált módszereket alkalmazzon, legyen alakítható és legyen tanítható.

1983-tól SSADM[128] néven kötelezően alkalmazandó lett minden új kormányzati és közigazgatási fejlesztés esetén. 1984-ben kibocsátották az SSADM 2-es változatát, 1986-ban pedig a 3-as változatot. 1987-ben a CCTA alapította Fejlesztés Felügyeleti Testület[129] vette át az SSADM gondozását. A 4-es változatot 1990-ben bocsájtották ki.

Elméleti alapozása Bachman (egyedek és kapcsolatok), Codd (relációk) és Jackson (feldolgozási és adatstruktúrák) munkásságára épül.

1988-ban hozták létre az SSADM-et oktató tanfolyamok minősítési rendszerét, a minősített tanfolyamok résztvevői eredményes vizsga letétele után SSADM szakértői oklevelet kaptak. 1991-ben kötelezővé tet-

125 L. 18. old.
126 Central Computer and Telecommunications Agency
127 Learmonth and Burchett Management Systems
128 Structured Systems Analyses and Design Method
129 Design Authority Board

ték ilyen szakértők alkalmazását az SSADM használatával megvalósítandó kormányzati informatikai projektekben. 1994-95-ben bejegyezték szabványként az Egyesült Királyságban.

1999-ben vizsgálták, hogy mennyire válik be az SSADM szoftverfejlesztési projektekben. 3 nagy SSADM projekt résztvevőit, 17 projektvezetőt kérdeztek ki és 90 felhasználót figyeltek meg betanításuk során. A vizsgálat eredménye az volt, hogy az SSADM nem képes jól megbirkózni a stratégia bizonytalanságával, a felhasználói kommunikációval és a személyzeti fejlesztéssel. A vizsgálatot ismertető tanulmányban azt a javaslatot fogalmazták meg, miszerint nagyobb figyelmet kell fordítani minden egyes projekt esetében a testreszabásra.[130] [F4339]

A CCTA, illetve az OGC[131] jelenleg ITIL[132] név alatt az információtechnológiai szolgáltatások menedzsmentjét támogatja mint a „legjobb gyakorlat"-nak megfelelő keretrendszert. Ez 2005-ben az információtechnológiai szolgáltatások menedzsmentjének első nemzetközi szabványává vált (ISO/IEC 20000). [F4343]

Az SSADM-mel Magyarországon az Információs Tárcaközi Bizottság 4-es számú ajánlása foglalkozik. Ezt az ajánlást az MTA Információtechnológiai Alapítvány, Kincses László és kollégái készítették még 1993-ban. [F4337] A Miniszterelnöki Hivatal honlapjára 1994 augusztusában került föl, ahol azóta is megtalálható a Kormányzati Informatikai Egyeztető Tárcaközi Bizottság[133] menüjében.

Az SSADM mint rendszerelemzési és fejlesztési módszer gondozását nem folytatják, így mivel az SSADM Felhasználók Csoportja[134] is megszűnt az ezredforduló után, elmondhatjuk, hogy mára sem társadalmi, sem kormányzati támogatás nem maradt mögötte. Ez azonban nem jelenti azt, hogy nem használható, vagy hogy ne lenne alkalmas célkitűzéseinek megvalósítására. A Budapesti Műszaki Egyetem Információtechnológiai Innovációs és Tudásközpontja például mindezek ellenére pont az SSADM-et választotta egy szoftver tervezéséhez 2006-ban. [F4363 p. 31.]

Jelenleg úgy tűnik, hogy az UML[135] válik fejlesztési ipari szabvánnyá, dacára annak a gyakorlati problémának, hogy túlságosan nagy és bo-

130 Vö. a Fejlesztési módszerek, módszertan c. rész (74. old.) utolsó gondolatával, a „Informatikusként örülünk..." kezdetű bekezdéstől.
131 Office of Government Commerce - Kereskedelmi Kormányhivatal
132 Information Technology Infrastructure Library
133 az ITB jogutódja, feladatait és összetételét a 1054/2004. (VI. 3.) Korm. határozat szabályozta újra
134 SSADM User Group
135 Unified Modeling Language

nyolult, és dacára annak az elméleti problémának, hogy nincs a formális nyelveknél megszokott és elvárt szigorú definíciója, szabványa (mint axiómarendszer) ellentmondásokat tartalmaz és nem teljes. Ráadásul hangsúlyozottan objektum-orientált, ami okozhat nehézségeket a hagyományos megközelítésű relációs modellezés-tervezés-megvalósítás során.

Fölmerül a jogos kérdés ebben a helyzetben, hogy van-e egyáltalán értelme még az SSADM-mel foglalkozni?

Véleményem szerint igen, több oknál fogva is. Elsősorban is a fejlesztési problémák legfőbb oka nem módszerbeli, nem tudásbeli, nem elméleti hiányosság, hanem a nem megfelelő modellezési-tervezési szemlélet. A siker szükséges feltétele a megfelelő modellalkotás, és a jó fogalmi szintű modellen alapuló logikai és fizikai szintű tervezés. Mivel ez szükséges (de önmagában még nem elégséges) feltétel, hiába akár a legjobb módszer, hiába akár a legjobb segédeszközök, az eredmény legalábbis kétséges. Ebből következik, hogy ha egy - bármely - fejlesztési módszert kiegészítünk a három szinten történő modellezés és tervezés követelményével, hatékonyabb módszert kapunk bármely más módszernél, amely ezt nem támogatja.

Az SSADM más fejlesztési módszerekhez - és a minőségbiztosításhoz - hasonlóan egy formális keretet biztosít, előírja, hogy mikor, milyen célból milyen dokumentumokat kell előállítani. Azok tartalmával, a tartalom minőségével azonban nem foglalkozik, talán nem is foglalkozhat. Az alkalmazó felelőssége, hogy a szabott kereteket a megfelelő minőségű tartalommal töltse meg. Ha az alkalmazó megfelelő minőségű tartalmakat állít elő, azaz átgondolt, alapos jó minőségű munkát végez, akkor a keret segítségére van abban, hogy lehetőleg semmit ne felejtsen ki. De a minőséget önmagában nem garantálja, nem tudja garantálni. Nagyon leegyszerűsített példával élve: ha a feleség elküldi a férjét a piacra, és fölírja neki, mi mindent kell vennie, a lista alkalmas eszköze annak, hogy a férj ne felejtse el valamelyik tételt beszerezni, de azt nem tudja garantálni, hogy mindenből a legfrissebbet, legjobbat és legolcsóbbat választja a férj.

Ezért tehát az SSADM is alkalmas példa arra, hogy a háromszintű modellezés-tervezés szükségességének és előnyeinek bemutatására keretet adjon. További megfontolásra érdemes szempont, hogy az SSADM módszer tartalmaz néhány olyan technikát, amely rendkívül hatékonyan segíti a modellalkotás esetleges hiányosságainak feltérképezését.

Az SSADM áttekintése

Az SSADM eljárási, műszaki és dokumentációs szabványok olyan gyűjteménye, amelyet kifejezetten a rendszerelemzés és a szoftverfejlesztés támogatására terveztek. Két fő részre tagolódik. Az első a követelmények elemzése, a második a rendszer tervezése. Nyílt rendszer, azaz használatához nem szükséges a CCTA engedélye. Támogatja a minőségbiztosítás (az ISO 9001 szabvány) bevezetését.

Számos olyan elemzési és tervezési technikája van, amelyek akár tőle függetlenül is jól használhatók, alkalmasak arra, hogy egymás eredményeit ellenőrizzék, következésképpen a végeredmény minőségét javítsák.

Az SSADM nem foglalkozik „balról" az informatikai stratégia kialakításával, tervezésével. Annak meglétét, pontosabban a rövid projektspecifikációk meglétét feltételezi. Ugyancsak nem foglalkozik „jobbról" a kivitelezéssel és a teszteléssel. Az SSADM a megvalósíthatósági elemzéstől a megvalósítás fizikai megtervezéséig terjedő részt fedi le. Ez a rész öt modulból áll. Az első a megvalósíthatósági elemzés, az azt követő három az elemzés, majd az azt követő egy a fejlesztés területe. Ezek rendre a követelményelemzés, a követelményspecifikáció, a logikai rendszerspecifikáció, majd a fizikai tervezés.

Úgy is jellemezhetjük az SSADM-et, mint amely megszabja, hogy a) mit, b) mikor, c) hogyan kell előállítani. A „mit" különböző dokumentumokat jelent, és az SSADM termékleírásai határozzák meg pontosan. A „mikor" meghatározása az SSADM strukturális modellje alapján történik: a fentebb felsorolt öt modult vagy fázist hét szakaszra osztja (ezeket 0-tól 6-ig számozza), a szakaszokat lépésekre bontja. Minden lépésnek pontosan meghatározott bemeneti és kimeneti dokumentumai vannak. A „hogyan" problémájára az SSADM számos technikája adja meg a választ.

Az SSADM egyik legfontosabb tulajdonsága, hogy rugalmas, azaz az adott (fejlesztési) körülményekhez igazítható, továbbá az egyik leghatékonyabb módszer, amely olyan szervezetek rendelkezésére áll, amelyeknek egy szabványos rendszerfejlesztési filozófiára és megközelítésre van szükségük. Az SSADM nyílt rendszer. Ez azt jelenti, hogy nyilvános, bárki számára hozzáférhető, bárki használhatja díjfizetés nélkül, engedélyt sem kell kérni a CCTA-tól.

A korábbi rendszerfejlesztési eljárások nem fektettek különösebb súlyt a végfelhasználó bevonására a döntésekbe. Szerepük a rendszerelemző ismeretekkel való ellátása volt, a specifikáció ellenőrzésében és a tesztelésben vettek, vehettek részt. Az SSADM ezen a területen na-

gyot lépett előre, mert jóval nagyobb szerepet szán a megrendelőnek: a fejlesztés további menetét meghatározó kritikus döntéseket ugyanis neki kell meghoznia.[136]

Három ilyen kritikus pont van: a megvalósíthatósági tanulmány során a *rendszer határainak kijelölése,* legfontosabb jellemzőinek meghatározása és a fejlesztési stratégia meghatározása során szükség van a megrendelő egyetértésére.

A rendszerszervezési alternatívák közül történő választás azt a célt szolgálja, hogy megalapozottan dönthessen a megrendelő a követelményeket kielégítő, különböző lehetőségek közül. A megrendelő érdekeit képviselők átveszik annak a felelősségét, hogy az új rendszer olyan szolgáltatásokat fog nyújtani, amelyek *találkoznak a felhasználók igényeivel.* Emellett ez a továbbfejlesztés számára szilárd alapot (mérföldkövet) biztosít.

A harmadik kritikus döntési pont a műszaki megvalósítás alternatívái. A megrendelő itt a lehetséges technikai megvalósítások közül választ, amely választás lényegében azt szabja meg, hogy a rendszer *hogyan, mi módon* fogja nyújtani azt a szolgáltatást, amelyet nyújtania kell.

Ez a három kritikus döntési pont nem pótolja és nem helyettesíti az adatmodellezés során nélkülözhetetlen, szoros és alapos együttműködést, viszont ennek során a megrendelő (és a felhasználó) nem hoz - kritikus fontosságú - döntéseket, hanem alapvetően tájékoztatást nyújt.

Az SSADM legfontosabb kitűzött céljai a felhasználói igényeknek megfelelő rendszer elkészítésének biztosítása határidőre és a tervezett költségkereten belül, a végtermék minőségének biztosítása, a fejlesztéshez szükséges erőforrások minél hatékonyabb felhasználása, a végtermék támogató eszközöktől való függetlenségének fokozása, illetve a rugalmasság növelése.

Szerkezeti felépítését tekintve öt modulból vagy fázisból áll, ezek hét szakaszra tagolódnak, minden szakasz több (változó számú) lépésből áll. Ezeket a lépéseket háromjegyű számok azonosítják, melyek első számjegye a megfelelő szakasz számát jelenti.

0. szakasz: Megvalósíthatóság

010. lépés: Felkészülés a megvalósíthatósági elemzésre
020. lépés: A probléma meghatározása

136 Vö. 31. old., Általános és egyedi nézetek

030. lépés: Megvalósíthatósági alternatívák kiválasztása
040. lépés: Megvalósíthatósági tanulmány összeállítása

KÖVETELMÉNYELEMZÉSI MODUL

1. szakasz: Jelenlegi helyzet vizsgálata

110. lépés: Az elemzés kereteinek megteremtése
120. lépés: A követelmények vizsgálata és meghatározása
130. lépés: Jelenlegi folyamatok vizsgálata
140. lépés: Jelenlegi adatok vizsgálata
150. lépés: A jelenlegi szolgáltatások logikalizálása
160. lépés: Elemzés eredményeinek összeállítása

2. szakasz: Rendszerszervezési alternatívák

210. lépés: Rendszerszervezési alternatívák meghatározása
220. lépés: Rendszerszervezési alternatíva kiválasztása

KÖVETELMÉNY SPECIFIKÁCIÓS MODUL

3. szakasz: Követelmények meghatározása

310. lépés: Igényelt rendszer folyamatainak meghatározása
320. lépés: Igényelt rendszer adatmodelljének kidolgozása
330. lépés: Rendszer funkcióinak előállítása
340. lépés: Igényelt adatmodell megerősítése
350. lépés: A specifikációs prototípusok kidolgozása
360. lépés: Feldolgozási folyamatok meghatározása
370. lépés: A rendszer-célkitűzések véglegesítése
380. lépés: Követelmények specifikációjának összegzése

LOGIKAI RENDSZERSPECIFIKÁCIÓS MODUL

4. szakasz: Rendszertechnikai alternatívák

410. lépés: Rendszertechnikai alternatívák kidolgozása
420. lépés: Rendszertechnikai alternatíva kiválasztása

5. szakasz: Logikai rendszertervezés

510. lépés: Felhasználói dialógusok meghatározása
520. lépés: Módosító feldolgozások tervezése
530. lépés: Lekérdező feldolgozások meghatározása
540. lépés: Logikai rendszerterv összeállítása

FIZIKAI TERVEZÉS MODUL

6. szakasz: Fizikai tervezés

610. lépés: Felkészülés a fizikai tervezésre
620. lépés: Fizikai adatszerkezet terv készítése
630. lépés: Funkció komponens háló elkészítése
640. lépés: A fizikai adatterv optimalizálása
650. lépés: A funkció specifikáció véglegesítése
660. lépés: A folyamat-adat kapcsolat véglegesítése
670. lépés: A fizikai terv összegzése

Fontos SSADM technikák

Az SSADM azon túl, hogy egy elég általánosan használható fejlesztési eljárásrendet szab meg, rendelkezik számos olyan technikai megoldással, amelyek igen hasznosak lehetnek akár az SSADM-en kívül is, mert igen hatékonyan segítik a helyes, vagy legalábbis a helyesebb adatmodell felépítését, megalkotását. Különösen fontosak azok, amelyek az esetleg hiányzó elemek földerítését segítik elő.

Ezek a technikák a folyamatmodellezés, a logikai adatmodellezés, a funkciómeghatározás, az egyed-esemény modellezés, ezen belül az egyed-élettörténet és az egyed-esemény mátrix, és végül, de nem utolsósorban a lekérdezési út.

Kétféle **folyamatot modellezhetünk**, esetenként kell is modellezünk. Szükséges modellezni a meglévő és működő rendszer folyamatait, emellett a tervezett, megvalósítandó információs rendszer folyamatait is terveznünk kell. Az utóbbiak nyilván nem teljesen függetlenek a meglévőektől, de egy új rendszer tervezése során mindig fölmerül, föl kell merüljön a kérdés, hogy a korábbi folyamatokat, a korábbi működést mennyiben lehet racionalizálni, egyszerűsíteni, kihasználva az újabb technika ígérte újabb lehetőségek előnyeit is. A folyamatmodellezés végterméke az adatfolyam-ábra (DFD, data flow diagram). Ennek alkotóelemei a külső entitás, az adattár, a folyamat (feldolgozás, pro-

cess) és az adatáram (vagy dokumentum-áram). Ezek felhasználásával jól modellezhetők akár a jelenleg működő folyamatok, akár a tervezettek. Lehetővé teszi a hierarchikus ábrázolást, következésképpen a hierarchikus tervezést úgy, hogy bármely folyamat (feldolgozás) tovább bontható a kívánt, ill. a szükséges részletességig.

A DFD-beli adattár nem szükségképpen feletethető meg egy az egyben valamely adatmodellbeli egyedtípusnak, általában azt mondhatjuk, hogy az adattár leginkább valamely nézet megfelelője lehet. A DFD ábra segítségünkre lehet a rendszer határainak megállapításában is. Érvényes ugyanis az a - logikus - szabály, hogy a feldolgozásokban adatok nem keletkezhetnek és nem veszhetnek el. Azaz a folyamatoknak legalább egy bemenő és emellett legalább egy kimenő adatáramuknak kell lennie. Ha ez a követelmény nem teljesülne, az vagy azt jelenti, hogy valamit nem vettünk figyelembe a modellezés során, vagy pedig azt, hogy az adott dolog mégsem folyamat, hanem külső entitás.

A folyamatmodellezés célja, hogy az adott információs rendszerről átfogó képet nyújtson, együtt ábrázolva a rendszer folyamatait és adatait. A rendszer határainak kijelölésén túl segítségünkre van a kimenő és bejövő adatáramok meghatározásában, a belső adatáramlások feltérképezésében, az adattárolás helyeinek meghatározásában, az adatok feldolgozását, átalakítását végző folyamatok meghatározásában.

A **logikai adatmodellezés** az SSADM folyamatának több pontján használatos: az elején a megvalósíthatósági elemzés során a meglévő rendszer és az igényelt rendszer adatmodelljének vázlatos áttekintésére, illetve legvégül a részletes és pontos adatmodell már a fizikai tervezés alapja. A „logikai adatmodellezés" név alatt az információs rendszer adatszerkezetének egyértelmű meghatározását értik. Ennek az elnevezésnek azonban nem sok köze van a fogalmi-logikai-fizikai szintű modellezés ill. tervezés középső eleméhez. „A technika használatával a szervezeti információ igények modelljét kell kialakítani, különböző részletességgel az egyes szakaszokban. A létrejövő adatmodell logikai, a szervezet működési szabályainak egyfajta statikus leképezése." [F4337 p. 171.] Itt több probléma is fölmerül.

Az adatmodellezés elsődleges célja ugyanis *nem* a szervezeti „információ igények" modellezése, hanem a szervezet valamely tevékenységi területét (területeit) írja le egyfajta módon (a fontos jelenségek, azok tulajdonságainak és kapcsolatainak feltárásával), azaz modellezi azt annak érdekében, hogy hatékonyabban lehessen azt működtetni. A hatékony(abb) működtetés természetesen jelentheti bizonyos információk megszerzését, előállítását is, bár ez a ritkább eset.

Egy példával élve: egy tanulmányi információs rendszer elsődleges alkalmazási területe az adminisztráció gépesítése az élőmunka-ráfordítás csökkentésének és az érintettek jobb kiszolgálásának érdekében. Ez nem zárja ki annak lehetőségét, hogy az időközben felhalmozott adatokat a rendszer üzemeltetője különböző szempontok alapján értékelje annak érdekében, hogy olyan új felismerésekre jusson, amely felismerésekre egyébként csak aránytalanul nagy ráfordítás árán, vagy esetleg egyáltalán nem juthatna.[137] Sajnos az utóbbi felhasználás a ritkább.

A másik probléma a „statikus" jelző. Az adatmodell ugyan tekinthető statikusnak a szó „nem változó" értelmében, hiszen ha az adatmodell állandóan, vagy legalábbis gyakran változna, az azt jelentené, hogy az adott terület működésében is alapvető változások álltak be (hiszen azt modellezzük), esetleg azt, hogy adatmodellünk az adott esetben nem elég jó, nem elég változástűrő. A statikus, „nem változó" modell azonban alkalmas arra is, hogy a modellezett rendszer üzemszerű időbeni változásait is le tudja írni.

A harmadik, legnagyobb probléma pedig az, hogy az itt említett logikai adatmodellezés a fogalmi modell és a logikai terv egyfajta keveréke, míg a fizikai terv részben logikai szintű elemeket is magába foglal. Magyarán: nem választja szét megfelelően a fogalmi, a logikai és a fizikai szintet, ami pedig megfelelő modellezés és tervezés formájában az eredményesség egyik előfeltétele lenne.[138]

Külön említést érdemel a **relációs adatelemzés** mint az SSADM által alkalmazott technika. „A relációs adatelemzés az SSADM-ben kiegészíti illetve ellenőrzi a logikai adatmodellezést. Egy második, teljesen eltérő nézőpontból vizsgálva a rendszer adatait a végső rendszertervet jobb minőségűvé tehetjük. A relációs adatelemzés az a technika, amelylyel az adatoknak egy olyan szerkezetét lehet előállítani, amely a lehető legkevesebb ismétlődést és a lehető legnagyobb rugalmasságot biztosítja (a szerkezet módosítása és kiegészítése terén). A rugalmasságot úgy lehet elérni, hogy az adatok csoportjait kisebb csoportokra bontjuk, az egyedi adatelemek közötti összefüggéseket figyelembe véve, az eredeti információtartalom elvesztése nélkül. (...) A fenti eljárást normalizációnak hívják" [F4337 pp. 242-243.]

A relációs adatelemzés véleményem szerint nem választható el az adatmodell kialakításától, annak kezdeti szakaszától sem. Célunk

137 L. A szelektív adatkezelés kihasználatlan lehetőségei c. részt, 110. old., ill. az Ismeretszerzési módok és csoportosítások c. részt, 45. old.
138 L. 26. old.

ugyanis a jó (sőt: a lehető legjobb) adatmodell kialakítása.[139] Ettől pedig elválaszthatatlan a normalizálás elvégzése. A szemantikus normalizálás[140] kapcsán írottaknak megfelelően csak ismételni lehet, hogy a normalizálást nem lehet pusztán mechanikusan végezhető technikának tekinteni, az nem választható el a modellalkotás folyamatától, a valósághű modell pedig egyben normalizált is.

Nézzünk ennek alátámasztására egy iskolapéldát: Hallgatók nyelvismeretét, pontosabban megszerzett nyelvvizsgáit kell nyilvántartani. Egy könnyen kínálkozó megoldás, hogy a hallgatókat leíró egyedtípusban szerepeltetjük a nyelvvizsgaeredményt: HALLGATÓ(HallgatóID, Hallgatónév, ..., Nyelv, Vizsgafok, Vizsgadátum). Könnyen felismerhető, hogy az utolsó három tulajdonságtípus ismétlődő csoportot alkot, hiszen lehetnek olyan hallgatók - ha kevesen is -, akiknek egynél több nyelvvizsgájuk van. Azaz egyedtípusunk normalizálatlan, normalizálni kell. Ennek módja a kivetítés: a helytelenül függő, illetve a nem függő tulajdonságtípusokat ki kell venni az eredeti egyedtípusból egy külön (esetleg már létező) egyedtípusba. Kapjuk tehát a HALLGATÓ(HallgatóID, Hallgatónév, ...,) egyedítpust és ennek alárendeltjeként a HALLGATÓ_NYELVVIZSGÁJA(HallgatóID, Nyelv, Vizsgafok, Vizsgadátum) egyedtípust.

Hasonló, sőt akár valósághűbb eredményre juthatunk a normalizálás formális eszköze nélkül is. A modellalkotás kezdeti szakaszában arra a felismerésre jutunk, hogy vannak hallgatók és vannak nyelvek, mindkettő saját jellemzőkkel leírható jelenség, tehát külön-külön egyedtípusokkal célszerű tükrözni azokat. A két egyedtípus egymással N:M kapcsolatban van, amit célszerűen föl kell bontani, a kapcsoló egyedtípus fogja leírni a hallgatók akármilyen nyelvekből megszerzett akárhány nyelvvizsgáit.

A **funkciómeghatározási technika** a funkciók leírására, a kapcsolódó bemeneti és kimeneti adatszerkezeteknek a definiálására irányul. A funkció olyan feldolgozási folyamat (vagy azok halmaza), amelyek a felhasználók számára elemi egységként jelennek meg. Ilyen például a hallgató számára a vizsgára való jelentkezés. Számára ez egy elemi művelet, ugyanakkor az elvégzendő ellenőrzések miatt igen sok táblát érintő számos lekérdezés van a hátterében. A funkciók a szükséges bemenetből, a bemenetre reagáló feldolgozási folyamatból vagy folyamatokból, továbbá az ezen folyamatok által létrehozott kimenetből állnak. Mint feldolgozási egységek a fizikai tervezés kiindulópontjaként szere-

139 L. 25. old.
140 L. a „Halassy Béla szemantikus normalizálásról ír...” kezdetű bekezdést, 38. old.

pelnek. A funkciók egy-egy futtatási egységgé válnak a megvalósítás során. Lehetséges csoportosításai szerint a funkciók lehetnek a) lekérdezők vagy módosítók, b) interaktívak vagy kötegeltek, c) a felhasználó vagy a rendszer által kezdeményezettek.

Az **egyed-esemény modellezés** az egyedek és az események egymásra való hatásának elemzése. Ez tekinthető talán a legsokoldalúbb ellenőrzési technikának, helye van bármilyen elv, módszer szerinti fejlesztési eljárásban. Segítségével számos modellalkotási hiányosság deríthető föl, azaz a modell teljességének biztosításában van fontos szerepe. Mind az egyedek, mind az események oldaláról vizsgálnunk kell a területet. Az első esetben kapjuk az egyed-élettörténet ábrákat (leírásokat), a második esetben az egyed-esemény mátrixot.

Az egyed-élettörténet elemzése során megállapítjuk az egyedeket befolyásoló eseményeket, a befolyásolás módját és sorrendjét, feltárjuk az aktualizálás peremfeltételeit és korlátait.

Az egyed-élettörténet feltárás során azt vizsgáljuk, hogy az egyes egyedekre hogyan hatnak az események. Az egyed-esemény mátrix az összes esemény által az egyes egyedekre gyakorolt hatásokat összegzi. Ez egy olyan táblázat, amelynek sorai a lehetséges eseményeket, oszlopai az egyes egyedeket jelentik. A mátrix celláiban a sorbeli eseménynek az oszlopbeli egyedre gyakorolt hatását tüntetjük föl. Ezek a hatások a következők lehetnek: létrehozás, módosítás, törlés (és az állapotjelző értékének változtatása). Azaz a mátrixból kiderül, hogy adott esemény bekövetkezte mely egyedtípusokban eredményezi új előfordulás létrejöttét, meglévő(k) valamely tulajdonságértékének vagy -értékeinek módosulását, illetve az előfordulás megszűnését.

A táblázat kitöltése segít földeríteni a hiányzó és a fölösleges eseményeket, illetve egyedeket. Ha ugyanis a táblázat valamely *sora üres,* az azt jelenti, hogy a sorbeli esemény egyetlen egyedre sem gyakorol hatást, tehát valójában vagy mint esemény fölösleges, vagy pedig, ha bizonyosan nem fölösleges, akkor adatmodellünk hiányos, mert nem szerepel benne egy vagy több lényeges egyed. Ha valamely oszlop üres, az pedig azt jelentené, hogy van olyan egyedünk, amelyre semmilyen esemény nincs hatással. Azaz lehetséges, hogy a szóban forgó egyed valójában fölösleges, vagy ellenkezőleg: eseményeink listája még hiányos.

Üres sor vagy oszlop előfordulása mégsem jelenti szükségképpen azt, hogy modellünk hiányos vagy hibás. Lehetséges ugyanis, hogy valamely egyedünk a rendszer eredeti célja és alapműködése szempont-

jából nem változó, csak olvasásra igénybe vett (pl. irányítószámok-települések).

A **lekérdezési utak** meghatározásának szerepe az adatmodell teljességének ellenőrzése. Minden lekérdezési művelethez, és minden módosító funkció lekérdező részéhez meg kell határozni a lekérdezési utat, azaz fel kell sorolni a lekérdezés során érintett egyedeket, olyan útvonalat kell kijelölni ki, amelyet egyszerű adatbázis-olvasási műveletekkel be lehet járni. Ez a technika az esetleges egyed- és kapcsolathiányokat segít meghatározni.

A legtöbb SSADM-technikát, használatuk módját személetes példákon kersztül, mondhatni élvezetesen mutatja be az Ismerkedés az SSADM-mel c. könyv. [F4300]

A modellezés és tervezés három szintje

Az SSADM, illetve az azt ismertető írások a módszer felépítésével kapcsolatban ismertetik a három nézet modelljét. Eszerint az elemzést három nézőpontból, a funkciók, adatok és események nézőpontjából végzi. [F4337 pp. 12., 17-19.] Ezen túl az SSADM legutóbbi, 4.2-es (4+) változatában megjelenik a három séma architektúra fogalma mint kulcsfogalom. [F4342 pp. 5-7.] Ez sajnos az ANSI-SPARC architektúra felfogását veszi át, amely az adatmodellezés szempontjából nem a legszerencsésebb.[141] Az SSDAM-nek - és minden más, konkrét rendszerek kifejlesztésére irányuló módszernek - a konkrét rendszerek adatmodellezését kellene hatékonyan támogatnia, míg az ANSI-SPARC architektúra - Codd három szintű architektúrájával egyetemben - a relációs adatbáziskezelők általános jellemzőiként értelmezhetők. Márpedig a sikeres és minőségi fejlesztés szükséges előfeltétele a lehető legjobb adatmodell kidolgozása. Ennek pedig a Halassy féle háromszintű modellezés-tervezés nélkülözhetetlen előfeltétele.[142]

Ismét a fogalmak pontos meghatározásának, illetve pontos használatának a problémájához kanyarodunk vissza.[143] Az SSADM, illetve az ANSI-SPARC szóhasználata szerinti fogalmi sémának semmi köze nincs a konkrét adatmodell fogalmi szintjéhez, hiába illetik ugyanazon elnevezéssel.[144] Ugyanez igaz a Codd által említett háromszintű megközelítéssel való összehasonlításra is.

A mindössze két szintre, logikai és fizikai szintre történő felosztás so-

141 L. 32. old.
142 L. 26. old.
143 L. az Alapfogalmak c. részben a tudomány nyelvéről mondottakat, 8. old.
144 L. 32. old.

rán a fogalmi szint formálisan mellőzött, gyakorlatilag részben, de csak részben beépül a logikainak nevezett szintbe. Ugyanakkor a logikai szint részben fizikai szintű elemeket is tartalmaz, vagy tartalmazhat. Ez azonban nem SSADM-sajátosság, hanem általánosnak tekinthető. Végső oka feltehetően az, hogy nem válik világosan szét a kétféle modellezési helyzet: más dolog az adatbáziskezelés relációs modellje és (nagyon) más dolog bármely adott információs rendszer adatmodellje. Márpedig a modellezésnél létfontosságú a modell érvényességi körének meghatározása. Ezért a kétféle „három szintű megközelítés": az ANSI-SPARC architektúra három szintje ill. Codd jóval pontosabb három szintje *nem* ellentéte vagy konkurrense az információs rendszerek adatmodellezése kapcsán elvárt és szükséges három szintnek. Egyszerűen másról van szó.

Már Codd három szintről beszél, ezek azonban szintén nem a fogalmi-logikai-fizikai szintet jelentik. „Három szintjét kell megkülönböztetni a fogalmaknak: 1) a pszichológiai szintet (a felhasználó szintjét), 2) a logikai és szemantikai szintet (logikai szint), 3) a tárolás és elérés módja (fizikai szint)"[145] [F4296 p. 461.] Ez a három szint sokkal inkább az ANSI három szintjére[146] hajaz, amely áll az ún. külső sémából (ennek Codd a nézetek szintjét felelteti meg), az elvi sémából (ennek Codd az alaptáblák szintjét felelteti meg) és a belső sémából (tárolás). [F4296 p. 34.] Tisztán kell azonban látnunk, hogy Codd alapvetően az adatbáziskezelés általános elveivel foglalkozik, a relációs kezelőmotor elvi alapjait fekteti le, és nem a felhasználói szemszögű adatmodellezéssel foglalkozik, így nem is volna tisztességes rajta számon kérni az adatmodellezés általunk elvárt szintekre történő felosztásának hiányát.

Peter Chen négy szemléleti síkot különböztet meg, amelyek közül kettő, a második és a harmadik kapcsolódik a relációs modellhez. Chen négy szemléleti síkja a következő: 1) a tudatunkban létező egyedekkel és kapcsolatokkal összefüggő ismeretek; 2) az az adatszerkezet, amelyben az egyedeket és a kapcsolatokat adatok valósítják meg; 3) az elérési módtól független adatszerkezet; 4) az elérési módtól függő adatszerkezet. [F4299 p. 10.]

Itt ismét hangsúlyozni szükséges, hogy Chen az adatbáziskezelés relációs modelljéről beszél, és nem konkrét helyzetek adatmodellezéséről, tehát az általa ismertetett szemléleti síkok nem összevethetők az

145 „Three levels of concepts must be distinguished: (1) psychological (the user's level), (2) logical and semantic (the logical level), and (3) storageoriented and access-method (the physical level)."

146 L. 32. old.

adatmodellezés három szemléleti síkjával[147]. Márpedig mindenképpen szükséges a fogalmi és a logikai szintet megkülönböztetni a fizikai szinttől, azaz (legalább) három fő szemléleti szinttel kell foglalkoznunk, nem tévesztve szem elől azt, hogy aligha létezik, aligha létez*het* a tökéletes fizikai adatfüggetlenség.[148] Emellett a technika túltengése, pontosabban a túlzottan technikai szemlélet, az eszközfüggőség okoz további gondokat.[149]

Emellett az SSADM-ben nemcsak hogy a fogalmi modellezés nem jelenik meg, hanem még a logikai és a fizikai tervezés is aránytalanul kis hangsúlyt kap, holott a végtermék elkészítése ezen modell, illetve tervek alapján történik, annak minősége közvetlenül ezektől függ! Éppen ezért az adatmodellezés, különösképpen a fogalmi modellezés jelentőségét és fontosságát nem lehet túlhangsúlyozni, ez pedig alkotó szellemi tevékenység, hasonlóan a tervezőmérnök munkájához, sőt kicsit a művész alkotómunkájához is. Nem lehet kizárólag mennyiségi tényezőként kezelni, különösképpen az időigény tervezésekor nem. Két ember nem fog tudni fele idő alatt - megfelelő minőségű - fogalmi modellt alkotni.

A különféle változatok nyomon követése és dokumentálása, a verizókövetés[150] is problémás, a módszer nem foglalkozik vele súlyának megfelelően. Ugyancsak háttérbe szorul a kódolási szabványok[151] kialakításának és alkalmazásának kérdésköre, amely a hatékony fejlesztésnek bár nem elégséges, de egyik szükséges előfeltétele.

Végül, de egyáltalán nem utolsósorban meg kell említeni az utógondozás hiányát is. Igaz ugyan, hogy az SSADM mint fejlesztési módszer a fizikai tervezéssel véget ér, tehát nem volna tisztességes számon kérni a tényleges szoftverfejlesztéssel, sőt a - majdani - üzemeltetéssel kapcsolatos előírások, elvárások hiányát. Azonban azt is figyelembe kell vennünk, hogy a feladat kierjedése az ötlet és az igény fölmerülésétől a használattal bezárólag tart. Ráadásul a fejlesztés, adott esetben a továbbfejlesztés és az üzemeltetés határai kissé elmosódottak, az üzemeltetéssel kapcsolatos előírások jelentős hányadát pedig a tervezés során lenne szükséges megalkotni, mert „...az adatbázis sikere nemcsak a korrekt terven, hanem a megfelelő használaton is múlik”. [F4260 p. 166.]

147 L. 26. old.
148 L. 30. old.
149 L. az Előtérben a technika... c. részt, 67. old.
150 L. 170. old.
151 L. 163. old.

A meglévő rendszer vizsgálata és elemzése rendkívül hasznos lehet a megoldandó probléma jobb megértése szempontjából. Evvel a 130., 140. és 150. számú lépések foglalkoznak, azaz három lépés. Ehhez képest a megvalósítandó rendszer adatmodellezése jelentősen a háttérbe szorul, hiszen a 320. lépésre (Igényelt rendszer adatmodelljének kidolgozása), illetve a 340. lépésre (Igényelt adatmodell megerősítése) szorítkozik csupán, azaz erősen a háttérbe szorul, fontosságához képest nem kap elegendő figyelmet. Adatintenzív rendszerek[152] esetében pedig elsődleges fontosságú az adatmodell, mégpedig a lehető legjobb adatmodell megalkotása. Helyesebb lenne ezen két lépésben a fogalmi szintű modellezést előírni, hiszen ennek elvégzéshez ekkor már minden lényeges ismeret rendelkezésre áll.

A logikai szintű adatszerkezet tervezésének pedig egyértelműen az 5. szakaszban van a helye. Ekkor már rendelkezésre áll nemcsak a fogalmi modell, de a 4. szakaszból azon körülmények is ismertek, illetve adottak, amelyek a logikai adatszerkezet kialakítása során figyelembe kell venni mint a hatékonysági, a technikai és hozzáférési követelményeket és/vagy korlátokat.[153]

A normalizálás helye és szerepe is megfontolás tárgya. „Az adatfolyam-ábrákat ezután formálisan meghatározott funkcióleírásokká és bemenet/kimeneti adatszerkezetekké kell alakítani. A logikai adatmodell érvényességét meg kell vizsgálni, illetve a tartalmát ki kell egészíteni a relációs adatelemzés, illetve az egyedtörténeti elemzés segítségével. Az eseményeket részletesen meg kell határozni, az eseményhatások elemzésének segítségével. Ezek illetve a lekérdezési utak[154] meghatározzák az adatelérési követelmények részleteit, alátámasztva a logikai adatmodellt.” [F4337 p. 88.]

A relációs adatelemzés, magyarán normalizálás a fogalmi szintű modellezés szerves része - kellene, hogy legyen.[155] A szemantikus normalizálásról már elmondottak alapján ez nyilvánvaló.

Egyéb észrevételek

További lényegi probléma az értéktartományokkal kapcsolatos álláspontja az SSADM-nek. „Bár a tartományok vizsgálata nem lényegi elem a relációk normalizálásában, az elemző azonosíthat és dokumentálhat fontosabb tartományokat az egyedi attribútumokhoz. Egy tartomány

152 L. az „Adatintenzív rendszerek...” kezdetű bekezdést a 123. oldalon.
153 L. a Logikai terv c. részt, 28. old.
154 L. 90. old.
155 L. a „Halassy Béla szemantikus normalizálásról ír...” kezdetű bekezdést, 38. old.

olyan értékkészletet jelent, amelyből az attribútumok aktuális értékeiket nyerik. Egy közös tartomány olyan érvényesítési és formátum beállítási szabályok, megengedett osztályok és értéktartományok összességét jelenti, amelyek egynél több attribútumra érvényesek." [F4337 p. 247.] Hasonló megfogalmazás olvasható az SSADM-et ismertető egyetemi jegyzet(ek)ben is, pl. [F4342 pp. 139-140.].

Az idézett rész két okból is problémás. Egyrészt úgy tűnik, mintha megfogalmazója nem lenne teljesen tisztában az értéktartomány (domain) jelentőségével, de még fogalmával sem. Codd azonban elsődleges fontosságúnak tartja az értéktartományok fogalmát a relációs adatbáziskezelésben.[156]

Az értéktartomány fogalmát az idézettnél pontosabban - pontosan - úgy lehet meghatározni, hogy „Az értéktartomány (domain) az adott jelentésű tulajdonság[típus] általánosan felvehető értékeinek a halmaza." [F4350 p. 56.] Lényegi eleme a meghatározásnak, hogy az *adott* tulajdonságtípus alaphalmazáról van szó. Ha a formális matematikai-halmazelméleti definíciókkal összevetjük, ez nyilvánvaló lesz, ugyanis a reláció formális, halmazelméleti definíciója is erre épül.[157] Ehhez képest tárgyi tévedés azt állítani, hogy a „közös tartomány (...) értéktartományok összességét jelenti, amelyek egynél több attribútumra érvényesek". Az értéktartománynak ugyanis *nem* lényegi eleme, hogy hány tulajdonságtípusra (attribútum) érvényes, már csak azért sem, mert gyakran csak és pontosan egyre. Ugyanis a fogalom pontos meghatározásából következik, hogy ha pl. a számlaszám értéktartománya a „két nagybetű után 9 számjegy" kitétellel leírható halmaz, és a rendelésszámaink véletlenül ugyanilyen szerkezetűek, a két tulajdonságtípus akkor sem alapozható a formai sajátosságok alapján egyformának tűnő halmazra. Mert gondoljuk el, holnap valamilyen körülmény megváltozása okán a számlaszámok szerkezete „három nagybetű után 12 számjegy"-re változik, a rendelésszám pedig maradhat a régi. Számlaszámoknak és rendelésszámoknak ugyanis semmi közük sincs egymáshoz, mert *különböző lényegek*.

Az értéktartományok ugyanakkor függetlenek a formai sajátosságoktól is, semmi közük sincs a „formátum beállítási szabályok"-hoz, mert az értéktartomány, az adott tulajdonságtípus lehetséges értékeinek halmaza, mindenképpen fogalmi szintű tényező, ellentétben a - pillanatnyilag - szükséges megjelenítési formával, amely logikai szintű tényező, és ugyancsak ellentétben a tárolási móddal, ami pedig egyértel-

156 L. az Értéktartomány c. részt, 60. old.
157 L. az Értéktartomány c. részt, 60. old.

műen fizikai szintű tényező.[158] A dátum jelentését, jelentésbeli lényegét nyilván nem érinti sem az, hogy hogyan jelenítjük meg (éé-hh-nn vagy nn-hh-éé), sem az, hogyan tároljuk (mondjuk unix időbélyeg formában).

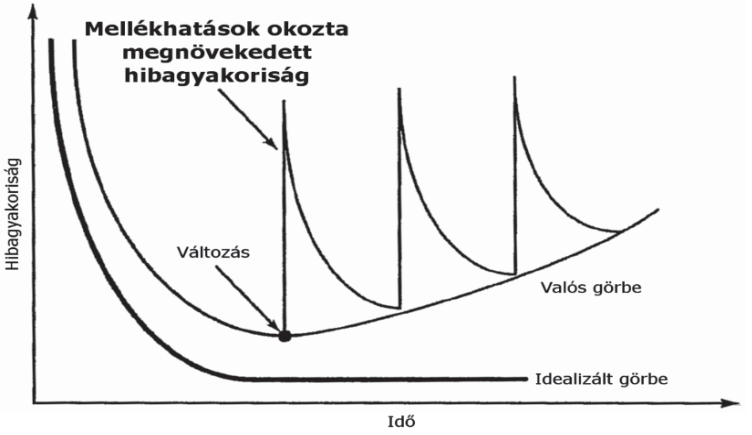
Az értéktartományoknak azonban nemcsak a matematikai-halmazelméleti definíció miatt van jelentősége. Szerepe van a normalizálás során is, mert vannak olyan helyzetek, amikor a funkcionális függés alapján nem lehet elvégezni a normalizálást, csak az értéktartományokon alapuló tartományfüggés alapján.

Szerepük van az adatbevitel ellenőrzése (érvényesítése) során. Amikor az adatbevitel során az adatot ellenőrizni kell, ennek legelső és kézenfekvő módja annak megvizsgálása, hogy a lehetséges értékek halmazából származik-e (számos egyéb megkötés is lehetséges ezen túl). Egy ember életkora pl. semmiképpen sem lehet nullánál kisebb, vagy - körülbelül - 120-nál nagyobb szám.

Az SSADM módszernek még felróható kisebb súlyú pontatlanság, hogy ugyan nem mondja ki egyértelműen és előzetesen, hogy relációs elvű adatbáziskezelést támogat, ugyanakkor azonban ezt egyértelművé teszi a relációs adatelemzés technika alkalmazása és beépítése. Ez azért érdekes, mert az adatbáziskezelés relációs elvének nem kellene elvárt követelménynek lennie. Igaz azonban, hogy a nem relációs elvű adatbáziskezelésre gyakorlati példa alig akad, és az sem bizonyos, hogy a relációs kezelők világában kialakult tervezési és fejlesztési technikák problémamentesen alkalmazhatók lennének gyökeresen más alapelven működő kezelők esetében.

A tervezés fontossága nem kizárólag az adatbázistervezés sajátossága. Általában igaz az, hogy a gondos, vagy gondosabb tervezés a későbbiekben meghálálja a ráfordítást, pontosabban a tervezés elnagyolása a kivitelezés és az üzemeltetés során bosszulja meg magát. A gondosabb modellezés, a gondos tervezés költségmegtakarító tényező. Könnyebb és olcsóbb a tesztelés, a javítás, az utánaigazítás, ugyanis kevesebb lesz a modellezési pontatlanság és a programozási hiba, illetve hiányosság, ideértve a kódolástechnikai és dokumentálási területeket is. Ez pedig közvetlenül és nagyon határozottan befolyásolja a végeredmény minőségét is. (6. ábra)

158 L. a Háromszintű megközelítés c. részt, 26. old.



6. ábra. A hibagyakoriság hatása a minőségre
(dr. Gyimóthy Tibor nyomán)

Ugyanis bármiféle változtatás, ideértve természetesen a fölismert hibák kijavítását is, további hibák lehetőségét eredményezi. Ebből viszont logikusan következik, hogy minél jobb minőséget sikerül elérni a fejlesztés során, egyrészt annál olcsóbb lesz a tesztelés és javítás, másrészt pedig a végeredmény minősége is - várhatóan - jobb lesz. Az ideális eset persze az lenne, ha rögtön tökéletes megoldás születne, ilyen azonban nincs. (Közismert szólás, hogy hibátlan program nincs, csak olyan, amelyet nem teszteltek elég alaposan.)

A program méretének és bonyolultságának kapcsolata exponenciális jellegű (a lineárisnál mindenesetre meredekebb) függvénnyel ábrázolható általános esetben. Ennek oka, hogy minél nagyobb méretű egy program, azaz minél több forrássort tartalmaz, annál problémásabb további sorok betoldása. Azt sem szabad figyelmen kívül hagyni, hogy adott, megkövetelt működést megvalósító pontosan definiált algoritmus is számos kódolástechnikai változatban készíthető el. Következésképpen annak valószínűsége, hogy egy program hibát, sőt hibákat tartalmaz, illetve a hibák mennyisége ugyancsak exponenciális jellegű (a lineárisnál mindenesetre meredekebb) függvénnyel ábrázolható a méret függvényében.

Manapság azonban mintha nem a minőségi munka, alapos és gondos modellezés és tervezés lenne az általánosan elterjedt, hanem mint-

ha egyre erőteljesebben érvényesülnének szakmailag elfogadhatatlan módon az üzleti megfontolások. Jelesül az ún. 80%-20% szabály és a „tedd rendbe később” elv (ha ezt egyáltalán szabad szabálynak, illetve elvnek nevezni).

A 80%-20% szabály az élet számos területén megjelenik, és azt fejezi ki, hogy az adott cél mintegy 80%-os készültségű szintjének eléréséhez szükséges ráfordítás az összes ráfordításnak mintegy 20%-a. Ebből viszont adódik az az üzleti szempontú megközelítési mód, hogy hozzávetőleg 80%-os (vagy nem sokkal magasabb) készültségi szinten érdemes átadni a „terméket”, aztán majd amikor nagyon muszáj, javítgatják. Sajnos, aki először alkalmazza ezt a megközelítési módot, nemcsak versenyelőnybe kerül (rövid távon mindenképpen), hanem a többi szereplőt is mintegy rákényszeríti, hogy így járjanak el, ugyanis a (bármely) termék árát az átlagos piaci ár szabja meg, és nem a tényleges ráfordítások. Napjainkban azonban nagyon nehéz az átlagos vevőt meggyőzi nemcsak a valódi minőség fontosságáról, de annak tényleges meglétéről is.

A „tedd rendbe később” felfogás gyakorlatilag ugyanezt jelenti, talán kicsit más megközelítésből. Lényege, hogy minél előbb, minél kevesebb ráfordítással használhatónak tűnő terméket produkál, ad át, üzemel be, mondván, hogy a fölmerülő problémákat majd később, utólag javítja. Másik, de ugyancsak hibás megközelítés, hogy „csak” a teljesítménnyel, a hatékonysággal nem foglalkoznak, azt elhalasztják későbbre, amikor maga a program már helyesen működik. Az ideológiai ízű megfogalmazás szerint előbb *működjön,* aztán működjön *jól, majd* legyen gyors.

Csakhogy nem szabad figyelmen kívül hagyni, hogy egy program(rendszer) helyes működésébe az is - logikusan - beletartozik, hogy *felhasználható* arra a célra, amelyre eredetileg szánták. Ez nemcsak azt jelenti, hogy befejeződik a futása és helyes eredményt ad, hanem azt is, hogy mindezt *megfelelő időn* belül teszi. Nem sokat ér ugyanis az a számítás, amely az időjárás minden korábbinál jobb 24 órás előrejelzését adja, de a futásideje 25 óra. Éppen ezért már a tervezés korai szakaszában kellő súlyt kell fektetni a teljesítményelemzésre és a kellő hatékonyság tervezésére.

Míg a céljának való megfelelőség nyilvánvaló követelmény, addig a hatékonyság általában sokkal kevesebb figyelmet kap, holott a teljes bekerülési költséget jobban befolyásolják a minőségi mutatók (pl. a hatékonyság), mint a funkcionalitás. Éppen ezért már a tervezési folyamat viszonylag korai szakaszában kellő figyelmet kell fordítani erre a

körülményre. [F4333 pp. 179-180.] Williams és Smith előbb hivatkozott tanulmányukban tízlépéses módszert ismertetnek ezen problémakör kezelésére. A tíz lépés önmagáért beszél: 1) folyamatok áttekintése; 2) architektúra áttekintése; 3) kritikus felhasználási esetek azonosítása; 4) kulcsfontosságú forgatókönyvek kiválasztása a kritikus esetekre; 5) hatékonysági célok azonosítása; 6) architektúra részletes tisztázása; 7) architektúra elemzése a hatékonysági célok szempontjából; 8) alternatívák azonosítása; 9) az eredmények, javaslatok bemutatása a vezetők számára; 10) gazdaságossági elemzés. [F4333 pp. 179-189.] [F4364]

A felhasználói igények

Adatbázisok és felhasználók

Az adatbázisok - egy lehetséges felosztás szerint - kétfélék lehetnek. [F4296 p. 6.] Vannak az úgynevezett termelésközpontú adatbázisok, ezek elsődleges feladata az ügyvitel gépesítése, míg az ún. kutatási adatbázisok esetében az elsődleges szerep az, hogy belőlük (segítségükkel) a lehetőségeket lehessen földeríteni (akár a jövőre vonatkozóan), illetve lehetséges jövőbeli tevékenységeket lehessen megtervezni. Ez párhuzamba állítható a tallózó és a szelektív ismeretkezelés fogalmaival.[159] Ez a csoportosítás azonban nem azt jelenti, hogy az adatbázisok általános érvénnyel besorolhatók a két - diszjunkt - halmaz valamelyikébe. Inkább felhasználási módot, pontosabban elsődleges felhasználási módot jelentenek. Ebben az értelemben egy tanulmányi információs rendszer elsődlegesen nyilván a termelésközpontú csoportba tartozik. Azonban benne van a *lehetőség* a tárolt adatok és azok összefüggéseinek formájában, hogy azok alkalmas elemzésével olyan új ismeretekre tehessünk szert, amelyeket egyéb úton egyáltalán nem, vagy csak aránytalanul nehezen lehetne megszerezni.

Nemcsak az adatbázisokat lehet csoportokba sorolni. Mint már fentebb szó esett róla, az adatbázisok, ill. az azokon alapuló információs rendszerek létrehozásának három fő résztvevője van: a (majdani) felhasználó, a fejlesztő és a vezető.[160] Azonban a felhasználók is többfélék lehetnek - egy lehetséges csoportosítás alapján. [F4314 pp. 70-76.]

Az egyik csoportba az ügyfelek tartoznak, mint végső felhasználók, míg a másik csoportba az ügyintézők sorolhatók (alkalmazási felhasználók). Azonban ha felhasználónak általános esetben mindazokat az

159 L. 45. old.
160 L. Általános és egyedi nézetek, 31. old.

embereket tekintjük, akik kapcsolatba kerülnek az ismeretekkel [F4314 p. 71.], akkor nem szabad figyelmen kívül hagynunk azokat sem, akik a rendszer létrehozásával, illetve később üzemeltetésével foglalkoznak.

Éppen emiatt mondhatjuk azt, hogy nem pusztán adatmodellt, nem pusztán jó adatmodellt, de optimális modellt kell alkotni a kezdeti stádiumban. Optimálisat, amely alapján mindkét felhasználási módban mindegyik felhasználó(fajta) számára az elérhető legjobb megoldás születhet meg.

Hallgatói tapasztalatok

Kérdőíves felmérést végeztem a Budapesti Műszaki Főiskolán[161] a hallgatóság körében 2008-ban. A kérdések értelemszerűen a főiskola által használt Neptun tanulmányi rendszerre vonatkoznak. A felmérés során összesen 9 kiegészítendő kérdésre kértem választ. A következő kérdéseket tettem föl:

1. A Neptun mely szolgáltatásait használod a leggyakrabban?
2. Mely szolgáltatásait tartod a legfontosabbnak?
3. Milyen szolgáltatásokat hiányolsz a Neptunból?
4. A Neptun mely szolgáltatásait tartod fölöslegesnek vagy nélkülözhetőnek?
5. Mi az, aminek (nagyon) másmilyennek kellene lennie?
6. Milyen rendszeres hibajelenséget tapasztaltál már?
7. Milyen véletlenszerű hibajelenséget tapasztaltál már?
8. Fordult-e elő adatvesztés? Ha igen, milyen körülmények között, és milyen (jellegű) adatok vesztek el?
9. Egyéb észrevételek?

Háromszáz kiadott kérdőívből 273 értékelhető volt. Ezek összesítése alapján a következő megállapítások tehetők. Általános benyomás, hogy a kérdőíveket kitöltő hallgatók jelentős hányada (kb. háromnegyede) nem gondolja át alaposan a kérdéseket és válaszait, hanem ötletszerűen, „kapásból” válaszol. Ez egyben előny is, mert így többnyire a legelső reakcióikat írták le.

Az első két kérdésre adott válaszok felsorolása a következő: pénzügy, órarend, üzenetek, tanulmányok (tárgyfelvétel, vizsgajelentkezés, tárgyleírások, eredmények). Az alábbi táblázatok megadják, hogy az adott választ a hallgatók hány százaléka jelölte meg (egy hallgató töb-

161 2010. január 1-től Óbudai Egyetem

bet is megadhatott, mindkét kérdésre átlagban 2,49-et jelöltek meg a felsoroltak közül).

1. táblázat
1. A Neptun mely szolgáltatásait használod a leggyakrabban?

pénzügy	54%
órarend	72%
üzenetek	69%
tanulmányok	54%

2. táblázat
2. Mely szolgáltatásait tartod a legfontosabbnak?

pézügy	54%
órarend	33%
üzenetek	59%
tanulmányok	56%

3. táblázat
A legfontosabbnak tartott szolgáltatások közül a tanulmányi részletesen

tárgyfelvétel	36%
vizsgajelentkezés	23%
eredmények	23%
tárgyleírások	21%

A legérdekesebb észrevétel az, hogy bár az órarend messze a leggyakrabban használt szolgáltatás, ugyanakkor ezt tartják a hallgatók a legkevésbé fontosnak.

A leggyakrabban használt szolgáltatások az órarend és az üzenetek (fogadása), gyakorlatilag holtversenyben (72% és 69%). Második helyen pontos holtversenyben a pénzügyek és a tanulmányokkal kapcsolatos tevékenységek állnak.

A fontosság szempontjából a pénzügy, az üzenetek és a tanulmányokkal kapcsolatosak eredményei igen közel állnak egymáshoz, míg az egyébként leggyakrabban használt órarendet tartják a legkevésbé fontosnak.

A tanulmányokkal kapcsolatos lehetőségek közül a legfontosabbnak tartott a tárgyfelvétel, a másik három közel egyforma súllyal szerepel. Számomra érdekes, hogy a vizsgajelentkezés lényegesen lemarad a tárgyfelvétel mögött, hiszen mind jellegüket tekintve, mind gyakoriságukat és (oktatói szemszögből legalábbis) fontosságukat tekintve egy-

forma súlyúaknak tűnnének. (Lehet, hogy a hallgatók sokkal jobban szeretik a tárgyfelvételt mint a vizsgázást?)

4. táblázat
3. Mit hiányolsz a Neptunból?

semmit	54%
tanárok elérhetőségét	15%
TO-ügyintézést	8%
ált. információkat, híreket	8%
több eredményt	8%
igazi levelezést	5%
jegyzetkeresést	3%

A 4. táblázatból három tételt emelnék ki. A hallgatók több mint fele úgy érzi, hogy a Neptun által nyújtott szolgáltatások elegendőek (54% semmit nem hiányol belőle). A tanárok elérhetőségét hiányolja a válaszadók 15%-a. Ennek érdekessége, hogy főiskolánkon majd' hat esztendeje szabványosították az emilcímeket, tehát az oktató nevének és karának ismeretében az emilcímet nagy biztonsággal tudni lehet. A kétnevű oktatók esetében egyértelműen, a jóval ritkábban előforduló háromnevű - többnyire férje és saját nevét is használó hölgykollégák - ez ugyan nem teljesen biztos, de nagy valószínűséggel megtippelhető. A főiskola honlapjának nyitóoldalán van egyébként telefonkönyv link, ahol mind a telefonszámot, mind a szobaszámot, mind az emilcímet meg lehet tudni - a név, vagy legalább alkalmas töredékének megadása alapján.

Érdekes, hogy az ügyintézést viszonylag kevesen hiányolják, akik viszont említik, azok azt vagy általában tették, vagy pedig nevesítetten az iskolalátogatási igazolásokkal kapcsolatban.

Feltűnő az „igazi levelezés” hiányolása, mégpedig azért, mert a hallgatónak amúgy is van (ha nem lenne: legyen) emilcíme, esetleg több is. A célszerűség pedig azt kívánná, hogy a levelezését mindenki egy helyen bonyolítsa, különben az esetleges visszakeresések nehézkessé válnak.

4. táblázat
4. Mely szolgáltatásokat tartod fölöslegesnek?

semmit	85%
véleményezés	5%
adategyeztetés	3%

jegykeresés	3%
egyes szolgáltatások összevonhatóak lennének	3%

A 4. táblázat adatai alátámasztják az előbbi eredményeket. Az igen kis arányban mégis fölöslegesnek tartott szolgáltatások esetében vélelmezhető, hogy magát a tevékenységet tartja fölöslegesnek a hallgató, és nem (csak) annak Neptunbeli megvalósítását. Kiemelnék kettőt: az adategyeztetést - bár elhanyagolható arányban - fölöslegesnek minősítők aligha gondolnak arra, hogy ha ezt nem „kényszeríti ki" a főiskola, annak a hallgatók látnák a kárát előbb vagy utóbb. Ennél fontosabb, jelentőségéhez képest méltatlanul alacsony arányban észrevételezték a hallgatók, hogy egyes szolgáltatások összevonhatóak lennének, pontosabban adott tevékenységet elegendő lenne egyetlen helyen és módon elvégezni tudni.

5. táblázat
5. Mi az, aminek (nagyon) másmilyennek kellene lennie?

semmi	21%
tárgyfelvétel és vizsgajelentkezés	21%
sebesség	13%
üzenetküldés nemcsak kód alapján	10%
időtúllépés	8%
stabilitás	8%
tárgykeresés	5%
egyszerűbb v. érthetőbb menüszerkezet	5%
ne kapjon érdektelen üzenetet	3%
gyűjtőszámla átfutási idő	3%
véleményezés	3%
zh-eredmények	3%
órarend	3%

A válaszadók egyötöde szerint minden rendben van, nem neveztek meg semmi változtatandót. Ugyanilyen arányban találták azt, hogy a vizsgajelentkezés és a tárgyfelvétel a problémás. A konkrétan megfogalmazott kifogások a sebességre (pontosabban lassúságra) vonatkoz-

nak, illetve az időtúllépési hibára, azaz hogy gyakran kilépteti őket a rendszer, nagy terhelések (tipikusan a vizsgajelentkezés és a tárgyfelvétel) eseten. Külön említi 13% a sebességet (lassúságot) mint problémát, illetve a stabilitással való elégedetlenséget fogalmazzák meg 8%-nyian.

6. táblázat
6-7-8. Rendszeres, véletlen hiba, adatvesztés

nincs	15%
lassú	10%
időtúllépés kicsi	26%
„kidob”	44%
adatvesztés	10%
lefagyás	10%
ömlesztett tárgylista tárgyfelvételnél	3%
„kérem várjon”	3%

Nem tapasztalt érdemi hibajelenséget a válaszadók 15%-a. A legtöbb észrevétel az indokolatlan kiléptetésekre vonatkozik, amely összefüggésben avval, hogy túl alacsony az időtúllépési küszöb (általában is, nagy terhelésnél pedig különösen), a lassúság és a lefagyás egyformán 10-10%-os arányban fordul elő. Bár az ún. „lefagyás” kliens oldalon nem írható kizárólag a kiszolgáló oldal számlájára, de annak szerepe van benne. Az együttesen messze legnagyobb arányú csoportot képviselő hibák a lassúság, a kicsi időtúllépés, az indokolatlan kijelentkeztetés az időszakonként jelentősen megnövekedő terheléssel (is) összefüggésben van. Méltatlanul alacsony arányban említik meg azt, hogy tárgyfelvételnél az összes lehetséges tárgy listáját megkapják, nemcsak azokat, amelyek kötelezőek, illetve amelyek közül választaniuk muszáj.

Az adatvesztés minden esetben elkallódott, vagy annak vélt üzenetet jelent. A „kérem várjon” jelenség nyilvánvalóan programkódolási hiba (oktatói felületen is előfordul, reprodukálható módon), lényege az, hogy adott helyzetben megjelenik a képernyő közepén egy ilyen feliratú panel, egyetlen OK gombbal. A „kérem várjon” üzenet tudomásul vétele (OK gomb) után viszont már semmit nem lehet csinálni az adott helyen, bármilyen további tevékenység a jelenség ismétlődését váltja ki.

Oktatói tapasztalatok

Oktatói felmérést nem készítettem, egyrészt kollégáim kímélése okán, másrészt pedig azért, mert a magam tapasztalatai is rendelkezésre állnak, mégpedig a kezdetek óta. Természetesen ez nem tekinthető tesztelésnek, és azt sem állíthatom, hogy mások nem tapasztalnak más problémákat, vagy nem értékelnek más jelenséget (esetleg hiányt) problémának, netán hibának. A legtipikusabb oktatói tevékenység azonban az eredmények beírása minden kolléga esetében. Mivel itt ugyanúgy tapasztalatokról van szó, mint a hallgatók megkérdezése esetén, rögzíteni kell, hogy ez nem tesztelési eredmény, a feltárt jelenségek nem tervszerű és módszeres tesztelés eredményei, hanem esetlegesek. Ezt előrebocsájtva az elmúlt években felgyülemlett észrevételek (hibák, hiányosságok) az alábbiak.

Általános észrevétel, hogy általában túl lassú, túl sok egérkattintást igényel az egyes feladatok ellátása (még ott is, ahol hatékonyabban lehetne akár egér nélkül, pusztán billentyűzettel megoldani valamit). Az időtúllépési hiba túl gyorsan következik be, és mindenféle jelzés nélkül. Egy évfolyam (150-200 fő) jegyeit még zavartalan körülmények között sem lehet emiatt egyszerre beírni, mert a névsor végére érve és menteni próbálva a beírt jegyeket, időtúllépésre hivatkozva a bejelentkező képernyőn találjuk magunkat. Ilyenkor a már beírt eredmények persze elvesznek.

Tételes hibaként sorolnám fel a következőket.

Különféle listákban a névsor a keresztneveket nem veszi figyelembe az abc-rendbe való besorolásnál, azaz előfordul, hogy pl. Kovács Pál megelőzi Kovács Andrást. Ennek oka az, hogy a keresztnevek láthatóan külön rovatba soroltak, viszont a lista elkészítésénél csak a vezetéknév oszlopra történik a rendezés, és nem a vezetéknév+keresztnév együttesre. A képernyőn történő jegybeíráskor nem feltétlenül ugyanaz a hallgatók sorrendje, mint a papírra a vizsga előtt kinyomtatott vizsgalapon.

Korábbi verzióban fennállt az a probléma, hogy a vizsgajegyet a kurzus felől (a vizsganap dátumára) megadva az nem jelent meg a vizsgaalkalom felőli eléréskor az eredmények között. Volt időszak, amikor a „nem jelent meg” esetet nem lehetett kezelni, mert a legördülő listából választható vizsgajegyek között ilyen lehetőség nem volt (és máshol sem). A vizsgaalkalomnál való jegybeírás esetén az alapértelmezett választás a vizsgajegy kellene, hogy legyen, nem az aláírás. Ugyancsak korábbi probléma volt, hogy a névsorokat alapértelmezetten neptunkód alapján rendezte a nevek helyett.

Egyszer vagy kétszer volt példa rá, hogy beírt jegyek eltűntek. Azóta a papíralapú listán feltüntetjük a neptunba való beírás időpontját és külön szignáljuk. Szerencsére jó ideje nem ismétlődött a jelenség.

Vizsgaalkalmak listájánál ha a kezdő- és végidőpontra megadjuk a (keresett) vizsgák egyetlen napját, üres listát kapunk, ha annál egy nappal hosszabbat adunk meg, akkor minden rendben (végdátumra nyilván nem „kisebb-egyenlő", hanem határozottan „kisebb" operátor használata).

Hallgató adott esetben állította, hogy az adott tárgy az ő fölvett tárgyainak listájában látszik, ugyanakkor az általam látott kurzuslistában a hallgató nem szerepelt, nem tudtam neki eredményt könyvelni. A problémát a tanulmányi osztály oldotta meg.

„Az alábbi hallgatóknak nem lehet aláírás típusú vizsgajegyet megadni, mivel a tárgykövetelmény nem ilyen típusú: N.N., X.Y.. Az alábbi hallgatóknak sikerült a jegybeírás: N.N., X.Y.." Ez az üzenet fából vaskarika, képtelenség.

Kurzusok/Jegybeírás, vizsgajegy törlési kísérlete során fordult elő a következő: „Hiba történt a törlés közben! Valószínűleg hivatkozás van valamelyik tételre. Ha vizsgajegyet szeretne törölni, akkor azt a vizsgajegybeírásnál teheti meg." A szóban forgó hallgatónak a kurzusok/jegybeírás felől látszott az adott napra könyvelt közepese, az adott napi vizsganévsorban viszont nem szerepelt.

A kurzus eredményeinek nyomtatása megjelenít ugyan egy nyomtatásra formázott(nak látszó) oldalt, de azt mentve a böngészővel ehelyett a táblázat helyett a megelőző oldalt menti.

2007 decemberében azt tapasztaltam, hogy a „saját" tárgyaim eltűntek, ugyanakkor nagyszámú, számomra idegen és ismeretlen tantárgy tárgyfelelőse lettem.

A hallgatói befizetéseket az oktatónak kellene ellenőriznie kurzusonként, külön menüpontban, ahelyett, hogy a rendszer azt maga ellenőrizné a vizsgajelentkezés egyik előfeltételeként.

Visszatérő probléma, hogy a vizsgakurzus hallgatóinak (akiknek már régebbről megvan az aláírásuk, azért is vehették föl a vizsgakurzust), nem engedi beírni a megszerzett jegyet, mondván, hogy még nincs aláírásuk. Muszáj az adott félévre még egy aláírást megadni nekik emiatt, ami nemcsak fölösleges, de ellentmondásos állapotot eredményez.

És néhány olyan észrevétel, ami ugyan nem hiba (legalábbis az adatbázistervezés, -üzemeltetés, programozás területén), viszont roppant bosszantó tud lenni - ergonómiai hibák.

Alapértelmezett módon egyetlen lapon kellene megjelenítenie a tel-

jes névsort (vizsgalapok, kurzusnévsorok), ugyanis ennek az adatmennyisége olyan csekély, hogy még alacsony sávszélesség esetén sem okozhat gondot, ellentétben a lapozás, vagy a lapozás kikapcsolásának időigényével.

Jegybeírásnál a dátum legördülő menüjében a hónapok neveinek az angol rövidítése jelenik meg.

Időnként előfordul a nehezen értelmezhető „ismeretlen hiba" üzenet.

Kurzusok/jegybeírásnál enged vizsgajegyeket beírni, azok elküldése után dobja vissza a hibaüzenetet, miszerint vizsgajegyet csak vizsgánál lehet beírni.

Kurzusok/jegybeírásnál az alapértelmezett jegyfajta a vizsgajegy, holott azt itt amúgy sem engedi beírni, és akkor is, ha a követelmény évközi jegy.

Oktatói: verziószám 288, 2007.01.19., 5:14:58 (2007.01.25-én du.)

Előfordul(t) nyilvánvalóan azonos képzés két különféle írásmóddal.

Üzemeltetői és adminisztrációs tapasztalatok

Körülbelül öt évvel ezelőtt voltam kari adminisztrátor, az akkori tapasztalataim és emlékeim jó része mára már nyilván túlhaladottá vált. Van azonban pár olyan észrevételem, amely messzebbre vezet a konkrét, egyedi hibák vagy kényelmetlenségek körénél.

A tantárgykódok mint „beszélő kódok" problémája. Az egyes tantárgyaknak van egy tárgykódjuk, amely eredetileg 10 karakter hosszú volt. Ennek felépítése a következő: kar (1), tanszék (2), tárgy (2), a tárgy hányadik féléve (1), „szabad" felhasználású (2), tagozat (1) és tanterv (1). Pl. GSVVI112NK jelentése: Keleti Károly Gazdasági Kar (G), Szervezési és Vezetési Intézet (SV), Vállalati információs rendszerek (VI), 1. félév (1), saját kari hallgatóinknak oktatjuk (1), Népszínház u-i telephelyen (2), nappali tagozatosoknak (N) és kredites tanterv szerint (K). Ez volt az eredeti elképzelés, azonban már a kezdet kezdetén problémák merültek föl. Kezdve azon, hogy a tárgy neve eleve Vállalati információs rendszerek 1., megkülönböztetendő a Vállalati információs rendszerek 2-től. Értelmezési, azaz modellezési kérdés, hogy mit is értünk egy tantárgy alatt,[162] de úgy tűnik célszerűnek, hogy egy tantárgy egyben egy féléves is legyen, tekintettel a félévre mint időbeosztási alapegységre. Az, hogy a példabeli Vállalati információs rendszerek 1. és 2. mennyire épülnek egymásra, sokkal inkább eldönthető az alapján,

162 L. Fogalomalkotás, 140. old., illetve A fogalmakban kifejeződő modellalkotás c. részt a 17. oldalon, továbbá az utóbbit közvetlenül megelőző bekezdéseket.

hogy a Vállalati információs rendszerek 1. előkövetelménye-e a Vállalati információs rendszerek 2. tárgynak, mintsem az elnevezésük alapján.

A tantárgykódokkal kapcsolatos probléma általános érvényű, az ún. beszélő kódok esetében előbb vagy utóbb mindig jelentkezik. A probléma pedig az, hogy mindig fölmerül olyan újabb körülmény, amelyet csak nehezen vagy egyáltalán nem lehet beleilleszteni az eredeti szerkezeti elképzelésbe. Esetünkben ilyen pl. a tárgy nevének kétbetűs rövidítése illetve a tanterv. Kezdetben „H" jelentette a hagyományos (értsd: nem kredites) tantervet, míg „K" az akkor új, kredites tantervet. Azóta azonban eltelt közel egy évtized, és már a sokadik kredites tantervváltozat van hatályban, és még lesz is egynéhány. Ráadásul már a kezdetekkor is volt egyik karunkon kreditrendszerű oktatás, ráadásul kétféle tantervvel, így már akkor frissiben adódott kivétel az általános szabály alól: az akkori kredites tantervek tárgykódjai 11 karakter hosszúak voltak, R illetve S betűkkel a legutolsó helyen.

A végső felhasználó, azaz a hallgató szemszögéből nézve a kérdést: mi szüksége van neki tárgykódokra? Azért van szüksége, mert részben annak megfejtése nyomán, részben pedig „szájhagyomány" útján tudja, hogy adott félévben mely tárgyakat is kellene felvennie. Ahelyett, hogy tárgyfelvételkor eleve csak azon tárgyak jelennének meg elsődlegesen felvehetőként, amelyek a mintatanterv előírásai, az addig általa teljesítettek (előkövetelmények) és a telephelyek (esetleg oktatók) alapján szóba jöhetnek. (Ezeken túl természetesen a jogszabályok és helyi szabályzatok szerint és alapján még azt választhat ki, amit akar vagy amit tud.)

A tantárgykódnak evvel együtt, pontosabban ezen túl megvan a maga szerepe és jelentősége. Az indexben feltüntetve jelentősen megkönnyíti az indexbeli adatok visszakeresését a számítógépes rendszerben.

A kezdeti időkben problémát okozott az, hogy az egyes tárgyakat kellett hozzárendelni az egyes képzésekhez, így nem lehetett ugyanazt a kurzust meghirdetni különböző karok hallgatói számára, ami komoly modellezési hiba, hiszen pl. az idegen nyelvek, a testnevelés és néhány más alaptárgy esetében is azok karfüggetlenek, ráadásul egy telephelyen három kar hallgatója is lehet(ett).

Ugyancsak komoly modellezési hiba, hiányosság, hogy nem volt lehetőség a felmentések nyilvántartására, egyszerűen senki nem gondolt erre a lehetőségre. Áthidaló megoldásként az született, hogy a felmentéseket a 8-as „érdemjegy" jelentette, amelyek viszont az átlagszámításban remélhetőleg nem vettek részt.

A hallgatói juttatások, így például a tanulmányi ösztöndíjak, rendszeres és eseti szociális támogatások folyósítása a Neptunon keresztül történt. Ennek módja az volt, hogy a megfelelő menüpont alatt ki kellett jelölni az ösztöndíjfajtát, és egy kötött formájú xls fájlban föl kellett tölteni a tényleges adatokat. A sikertelen átutalásokat (pl. mert a hallgató elfelejtette időben jelezni, hogy megváltozott a számlaszáma) sztornózni kellett. Ez utóbbi volt a nagyon problémás, ugyanis a sztornózás ablakban ki kellett választani a sztornózandó ösztöndíj fajtáját, az átutalás dátumát, megadni a hallgató kódját és adatbeviteli mezőben(!) az összeget. Hiba esetén javításra mód nem volt. Figyelembe véve a sztornózás jelentését, a már elküldött tételek (szűkíthető) listájából lehetne csak kiválasztás útján kijelölni a sztornózandó tételeket. Ráadásul az ösztöndíjfajták legördülő menüje jóval keskenyebb volt mint az ösztöndíjfajták elnevezései, számos esetben találgatásra késztetve a rendszergazdát.

Máig ható problémák forrása a redundancia. A Neptun adatbázisa főiskolánkon kb. 40 GB (!) méretű, és az egyik kari tanulmányi osztály vezetője sajnálkozik afölött, hogy bár a jogosultsága meglenne hozzá, de mégsem tud érdemben SQL-lekérdezéseket futtatni az adatbázison, mert mindig bizonytalan abban, hogy az az adat, amelyet felhasználna, a valódi-e. Számos esetben ugyanis csak az óvta meg a melléfogástól, hogy a kapott listákról egyéb ismeretei folytán ismerte föl, hogy hibásak.

Sokkal, de sokkal könnyebb egy létező és használt megoldást vizsgálni, elemezni, kritizálni mint azt megtervezni és megvalósítani. Éppen ezért fontosnak tartom kiemelni, hogy itt *nem* a Neptun „lejáratása”, „kicikizése” a célom, hanem mint a meglévő két (ismertebb) megoldás egyikét tudtam valamelyest vizsgálni. Ez a vizsgálódás messze van attól, amit teljesnek lehetne mondani, nem tekinthető tesztelésnek, pusztán összegyűjtött esetleges tapasztalatoknak, mégis fontos ezt megtenni a tapasztalatok összegzése és a hasonló problémák elkerülése céljából. Emiatt tekinthettem el attól, hogy tanártársaimat, tanszéki adminisztrátor, tanulmányi ügyintéző és rendszerüzemeltető kollégáimat részletesen és alaposan kikérdezzem.

Egyre könnyebb használhatóság

Ahogyan egyre több információs rendszer felhasználói leszünk, úgy gyűlnek általános tapasztalataink is, úgy válik egyre inkább alapismeretté az „információs társadalomban” való hatékony és célszerű eligazodás. Evvel párhuzamosan egyre markánsabban megfogalmazódik az

az igény, hogy az egyes esetekben egyre könnyebben, kényelmesebben, hatékonyabban használható megoldásokat szeretnénk. Teljes joggal.

Esetünkben egy tanulmányi rendszer elsődlegesen termelésközpontú[163], elsődleges célja az ügyvitel gépesítése, annak hatékonnyá tétele. Ebből következően az ügyfelek és az ügyintézők lehetséges szempontjait kell elsősorban érvényesíteni. Ennek kapcsán az említett kétféle felhasználó között markáns különbséget kell tennünk abból a szempontból, hogy míg az ügyféltől (végső felhasználó, a hallgató) az általánosan elvárható számítástechnikai jártasságot várhatjuk el, addig az ügyintézőtől (alkalmazási felhasználó) azt is elvárhatjuk, sőt megkövetelhetjük, hogy az általánosan elvárhatónál nagyobb jártassága legyen, magyarán a célszerű és hatékony használat munkafogásait megtanulja.

Mindkét csoportba tartozó felhasználó számára egyformán fontos, hogy a leggyakoribb műveleteket a legkönnyebben és a leggyorsabban, illetve a legegyszerűbben lehessen elvégezni.

Bármely tevékenység, művelet a lehető legegyszerűbben legyen elvégezhető, esetleg többféleképpen is (ügyintézői körben esetenként kötegelt mód lehetősége), azaz legyen könnyen kezelhető, természetes logikát követő. Ebbe az is beletartozik, hogy az adatbázisban amúgy is rendelkezésre álló adatok (pl. különféle kódok) ismeretét nem szabad elvárni, megkövetelni a felhasználóktól. Ez azonban nem zárja ki annak lehetőségét, hogy adott esetben különféle kódértékek ismerete meggyorsíthassa a munkát.

De lehetne kezeléstechnikai példát is mondani: ha egy oktató jegyet ír be egy hallgatónak, akkor ne kelljen pl. egy külön egérkattintást tenni csak azért, hogy megjelenjen a rovat sarkában az a gomb, amelyre másodikat kattintva legördül a választható jegyeket tartalmazó lista, amelyből egy harmadik kattintással lehet a jegyet kiválasztani.

Általános érvényű, sokoldalú felhasználhatóság

Közhelyszerű, de még mindig nem kellőképpen tudatosított kijelentés, hogy az adatbázis erőforrás. Mint ilyen, önmagában véve is érték, azaz pénzt ér - és pénzbe kerül. Üzleti szempontú megközelítésben az az értelemszerű cél, hogy ebből az erőforrásból (is) a lehető legtöbbet lehessen kihozni, a lehető legnagyobb nyereség elérését segítse - nem feltétlenül közvetlen módon.

Mivel a számítástechnika, ezen belül is a relációs adatbáziskezelés

163 L. az Ismeretszerzési módok és csoportosítások c. részt, 45. old.

forradalmian új lehetőségeket teremtett, meggondolandó, hogy ezen lehetőségek maradéktalan kihasználása akár forradalmian új lépéseket tehet szükségessé.[164] „Ebben az átalakulásban alapvető dimenziók határozzák meg az innovációs projektek sikeréhez vezető cselekvéseket: a) a menedzsment dimenzió a stratégiaalkotást, a pénzügyi feltételek biztosítását, az irányítási módszereket és ezek magas színvonalát jelenti, b) a végrehajtási dimenzió az eljárásokat, a folyamatokat, az értékelőállítást, a fogyasztói igények kielégítését, a csoportmunka támogatását, c) a szociális dimenzió a szervezeti és humán aspektus, amelyet mindenféle szervezeti átalakításnál, újjászervezésnél, technológiai váltásnál figyelembe kell venni, és végül d) a technikai dimenzió a szervezeti feladatok végrehajtását támogató informatikai és egyéb technológiai elemeket definiálja.” [F4336 p. 112.]

Éppen ezért nem elegendő, nem megfelelő megoldás a történelmileg hagyományos ügyviteli folyamatok mechanikus módon történő gépesítése, hanem minőségileg új megoldásokat és formákat lehet, sőt kell kialakítani, amelyek jobban megfelelnek egyrészt az igényeknek, másrészt pedig az amúgy is rendelkezésre álló lehetőségeknek. Figyelembe véve azt, hogy az oktatás, ezen belül a felsőoktatás nem tekinthető közönséges vállalatnak a szó üzleti-gazdasági értelmében, úgy gondolom, hogy a radikális, forradalmi átalakításokat a lehető legszűkebb területre kell szorítani annak kockázatai miatt, és a folyamatos, fokozatos minőségfejlesztést érdemes előnyben részesíteni. Azaz a BPR[165] helyett inkább a TQM[166] részesítendő előnyben. Természetesen figyelembe véve nemcsak a terület, de az egyes elemek sajátosságait is, mert vannak dolgok, amelyekre nem lehetséges fokozatosan áttérni. Ilyen például a kreditrendszer bevezetése. Viszont számos - a legtöbb - más területen lehet szinte napról-napra javítani. Erre a későbbiekben javaslatot is teszek a pénzügyek területén.

A szelektív adatkezelés kihasználatlan lehetőségei

Valamennyien megfigyelhetjük, hogy hallgatóink motivációjának, tudásának és szorgalmának átlagos szintje évről-évre csökken. Hallgatóink jelentős hányada arra törekszik, hogy energiaminimumon megsze-

164 L. A modellalkotás szerepe c. rész elejét, 18. old.

165 Business Process Reengineering - vállalati folyamatok újraformálása - a szervezeti struktúra és folyamatok radikális átszervezése, megváltoztatása

166 Total Quality Management - teljeskörű minőségirányítás - fokozatosan, de folyamatos módon történő szervezetfejlesztés

rezze a diplomát, ahelyett, hogy a maximális tudás megszerzésére törekedne.

Ha a felsőoktatásban részt vevő hallgatók száma növekszik, az egy diplomára jutó IQ és tényleges tudás csökkenni fog. Ezen folyamatnak nemcsak az az oka, hogy alacsony szintű a társadalmi tőke befektetése magába a társadalomba, hanem a hallgatói létszám növekedése is. Nagyszámú hallgató esetén az értelmi képességek mennyisége, helyesebben mondva értéke Gauss-eloszlást követ. Ritka kivételektől eltekintve az érettségizetteknek a legjobb része folytatja tanulmányait a felsőoktatásban, ami azt jelenti, hogy őket a Gauss-görbe jobb oldalán ábrázolnánk. A hallgatói létszám növekedése szükségképpen azt jelenti, hogy a növekmény a Gauss-görbe bal oldalának irányából érkezik, azaz alacsonyabb szellemi képességekkel bír. Az elégtelen társadalmi tőkeberuházás és a létszámnövekmény együttesen azt eredményezik, hogy a lemaradó, sőt kimaradó hallgatók aránya növekszik.

Hogyan lehet az informatikát ezen folyamat lassítására, jobb esetben megfordítására használni? Úgy, hogy információt fektetünk be mint egyfajta társadalmi tőkét. Az alapvető erőforrás nem a tőke, nem a természetes erőforrások, nem a telek és nem is a munkaerő, hanem a tudás." [F4336 p. 112.][167]

A felsőoktatásban lassan egy évtizede használnak számítógépes tanulmányi nyilvántartást. Van tehát elég nagy mennyiségű adatunk múltbeli és jelenbeli hallgatóinkról. Ez az adattömeg megadja az információhoz jutás lehetőségét annak értelmezése útján. Ezen szempontból a számítógépes adatbázisok előnye az, hogy az adatok bármilyen módon válogathatók és csoportosíthatók pillanatok alatt. Ezen adatok pedig jól meghatározott, régen bevált statisztikai elemzések segítségével vizsgálhatók.

Vizsgálhatjuk például, hogy van-e bármiféle kapcsolat a különböző tantárgyakból elért eredmények vagy adott tárgy és annak előkövetelményeiből elért eredmények között. Kereshetünk összefüggéseket a tanulmányi adatok, a társadalmi környezet és a motiváció között. Nyilván az utóbbiak egyéb adatforrásokat is igényelnek.

Megelőző programokat lehet indítani azon hallgatók számára, akik esetében nagy a valószínűsége annak, hogy nem fogják tudni befejezni, vagy időben befejezni tanulmányaikat. Ez csökkentheti a lemorzsolódók arányát akkor, amikor a lehetséges hallgatók létszáma amúgy is csökken, ráadásul a finanszírozás létszámarányos.

167 Drucker P.: Post-Capitalist Society - Harper Business New York, idézi Raffai.

Stratégia tervezés

A modellalkotás szerepéről szóló részben már idéztem Codd véleményét, aki Einsteinre hivatkozva ragaszkodott a célszerű egyszerűséghez.[168] A modell legyen olyannyira egyszerű, amennyire csak lehet, természetesen az elérendő cél figyelembe vételével, azaz tartalmazzon minden szükséges elemet, de azon túl semmit. A minimalitás követelménye nemcsak a redundanciától való mentességet jelenti, de azt is, hogy a kitűzött cél elérése szempontjából nélkülözhető elemek egyben nélkülözendők is.

„A fejlesztők azonban bíznak benne, a gigantikus informatikai struktúrák előbb-utóbb belopják majd magukat a diákok és tanárok szívébe – például akkor, amikor már nemcsak vizsgára jelentkezhetünk majd az ETR-en, de a csoporttárs által készített puskát is letölthetjük róla." [F4288]

A két álláspont homlokegyenest ellentmond egymásnak. Véleményem szerint Coddnak van igaza. A hatékony működés, üzemeltetés, sőt fejlesztés előfeltétele, hogy minél egyszerűbb és kisebb legyen a rendszer - természetesen a lehetőségek határain belül, azaz úgy, hogy a szükséges feladatokat maradéktalanul el tudja látni. Nem célszerű olyan feladatokat, szolgáltatásokat nyújtani, amelyeket más eszközök történelmileg hagyományosan, magas színvonalon megvalósítanak. Így például különösen nem érdemes elektronikus levélküldéssel, vagy puskák (és egyebek) fel- illetve letöltögetésével terhelni egy ügyviteli rendszert.

A fenti értelemben vett egyszerűség követelményéből következik, hogy a tervezendő rendszer által nyújtott szolgáltatások körét is az észszerűség határain belül a lehető legszűkebben kell, illetve érdemes meghatározni. A modell jóságának általános értelemben vett előfeltétele, hogy az adott cél szempontjából szükséges *és* elégséges tulajdonságokat vegyük figyelembe.[169]

Információs rendszer kialakítása előtt, illetve annak kapcsán mindig fölmerül az a kérdés, hogy a számítógép-alapú rendszer üzembeállítása mennyiben teszi lehetővé, illetve szükségessé a meglévő ügyviteli rend, vagy akár a szervezeti struktúra átalakítását.

Erre a kérdésre a lehető legrosszabb válasz az, ha semmilyen válasz nem születik, azaz a számítógép-alapú rendszer sajátosságait és lehe-

168 L. 20. old.

169 L. az állapotjellemzésről írottakat a Modell, modellezés, hasonlóság c. részben, 15. old.

tőségeit teljes mértékben figyelmen kívül hagyva marad minden a régiben. Ez értelmetlen, sőt káros. A számítógép(ek) használatának pont az a legfőbb oka és értelme, hogy alkalmazásukkal emberi munkát, munkaidőt és fáradságot lehet megtakarítani, a nagy mennyiségű, gépies feladatok gyors és hatékony gépi elvégeztetésével.

Az érdemi válasz két fő lehetőség valamelyikét kínálhatja: a radikális, alapvető, gyors változtatásokat (BPR) és a minőséget folyamatos javítás útján emelő megoldásokat (TQM), illetve ezen lehetőségek értelemszerű kombinációit.

Mind a BPR, mind a TQM nagyon jó eredményeket hozhat, amennyiben alkalmazásának feltételei fennállnak, és a folyamatot sikerül végig is vinni. A szakirodalom szerint azonban az esetek közel felében ez az eredmény nem következik be sem a BPR, sem a TQM esetében. [F4340 p. 19.]

Figyelembe véve a felsőoktatás sajátos helyzetét, általános módszerként a TQM javasolható, kivéve az olyan helyzeteket, amikor az elvileg lehetetlen (l. pl. a kreditrendszer vagy a BSc-MSc rendszer bevezetését).

A relációs adatbáziskezelők többnyire rendkívül érzékenyek a lekérdezések megszerkesztésére. Ugyanazon válaszlistát eredményező, tehát lényegileg azonos, formailag azonban különböző lekérdezések futásideje között akár nagyságrendi különbség is lehet. Szükséges tehát fokozottan odafigyelni az SQL parancsok felépítésére, szerkezeti kialakítására is. Ennek alapvető módszereit, lehetőségeit mutatja be pl. Ullman. [F4361 pp. 268-316.]

Mivel egy tanulmányi rendszert (nagyon) sok féléven keresztül használnak normális esetben, ezért a fejlesztésre fordított erőforrásokkal a minőség rovására takarékoskodni elhibázott dolog lenne. Különösen vonatkozik ez a minőségi adatmodellezésre, mint a jó minőségű végtermék talán legfontosabb előfeltételére. Ez azonban csak akkor reális elvárás, ha az üzleti-anyagi szempontokat sikerül nagyrészt (vagy teljesen) kiiktatni. Erre kiváló lehetőség a szabad szoftver.[170]

A szoftverkrízis[171] zsákutcáit elkerülendő az alkalmazott módszernek meg kell felelnie az általános technikai elvárásoknak, azaz a célszerűség, a gazdaságosság, a tervszerűség és a szervezettség követelményének. [F4289 p. 45.] Mivel a hibák előfordulásának gyakorisága a programméret függvényében exponenciális (jellegű) növekedést mutat,

170 L. a „Szabad szoftvernek...” kezdetű bekezdéstől a Technikai, emberi és szervezeti környezet c. részben, 137. old.

171 L. az Elvárások - szoftverkrízis c. részt, 121. old.

létkérdés a feladat a minél kisebb, egymástól független programmodulokra való felosztása.

Összefoglalva: a hatékonyságot befolyásoló fő tényezők, fontossági sorrendben: a) az adatmodell egyszerűsége, b) az okvetlenül szükséges szervezeti átalakítások elvégzése, c) megfelelő logikai és fizikai tervezés. Stratégiai célkitűzés tehát a megvalósítandó rendszer minimalitása, azaz szolgáltatási körének a feltétlenül szükséges körre való korlátozása, az ügyviteli folyamatok közül a legszükségesebbek értelemszerű átalakítása, a futásidőre, illetve egyidejűleg kiszolgálható felhasználószámra való optimalizálás a logikai-fizikai szintű tervezés és a megvalósítás során.

Módszertan választás, ill. kialakítás szempontjai

Az életciklus modell[172] legnagyobb hiányossága az, hogy feltételezi az igények pontos megfogalmazását, márpedig a projektindításkor általában még jelentős mértékű bizonytalanságok szoktak lenni az elvárások és feltételek vonatkozásában. [F4282 p. 315.]

Esetünkben azonban pont ez a probléma nem áll fenn. Az oktatási folyamatok, azok feltételei és az azokkal szemben támasztott elvárások, a teljes ügyvitel pontosan ismert és szabályozott. A kreditrendszerre való áttérés jelenthetett ebből a szempontból bizonytalanságot, de ez már évekkel ezelőtt megtörtént, a szükséges tapasztalatok rendelkezésre állnak, azok beépítése az ügyviteli rendbe megtörtént.

A választandó, illetve kialakítandó fejlesztési módszer vagy módszerek kiválasztásánál (kialakításánál) a következő, általános szempontokat lehet megfogalmazni: a) problémához illeszthetőség; b) minősítés; c) időtényező; d) általános érvényűség. [F4282 p. 333.]

Azaz nagyon fontos odafigyelni arra, hogy a módszer ne csak az adott probléma megoldására legyen általában alkalmas, de illeszkedjen a meglévő, adott fejlesztési és üzemeltetési környezetbe. A módszernek azonban nemcsak a problémához kell illeszkednie, hanem azt is tekintetbe kell venni, hogy a fejlesztési módszer hatékonyságát *nem* a fejlesztés sikere vagy bukása jelenti, hanem az elkészült termék *minősége*. Mivel utóbbi elsősorban emberi tényezőkön múlik, a módszernek olyannak kell lennie, amelyet a résztvevők biztonsággal és hatékonyan tudnak - és akarnak - alkalmazni. Nem szabad figyelmen kívül hagyni, hogy egy módszer eredményességét általában csak nagyobb időtávlatban lehet megítélni. Mivel kevés olyan módszer van, amely bármiféle

172 L. az SSADM módszertani problémái c. rész elejét, 73. old.

problémára egységesen alkalmazható lenne, ezért a fejlesztés különböző fázisaiban esetleg különféle módszereket lehet érdemes alkalmazni. Nem szabad beleesni a szolgai módon történő alkalmazás csapdájába, az emberi kreatív, problémamegoldó gondolkodás elsőbbséget élvez.

A konkrét helyzetben olyan módszerre, vagy módszerekre van szükség, amely a fenti általános követelményeken túl a teljes életciklust átfogja a probléma fölvetésétől az üzemeltetésen keresztül a karbantartásig. Ezen túl lehetővé teszi a környezeti folyamatok szükség szerinti átalakítását. Végül, de nem utolsósorban pedig megteremti, sőt kikényszeríti a jó minőségű végtermék szükséges (de önmagában nem elégséges) előfeltételeként a minőségi, háromszintű adatmodellezést. Azaz a fogalmi modellezésen alapuló logikai, majd fizikai tervezést.

Ismét hangsúlyozni kell, hogy fejlesztési módszer választásáról van szó, és nem projekttervezésről, -vezetésről. Olyan fejlesztési módszert kell választani, vagy kialakítani (testre szabni), amely illeszkedik a teljes projektfeladat tervezési és végrehajtási rendjébe.

Fejlesztési módszerként az SSADM módszert veszem alapul mint olyant, amely már kipróbált és bizonyított is. A módszer tagadhatatlanul meglévő hiányosságai közül véleményem szerint a legfontosabb az adatmodellezés nem kielégítő támogatása. Ezért javasolom bővíteni a fogalmi szintű adatmodellezéssel.

Tervezési segédeszközök választása

Tervezéshez használható segédeszközt választani nem könnyű dolog, pedig rengeteg van. Raffai Mária már idézett művében fölsorol 35-öt[173], de pl. a City University of New York „CASE tool index” címen több mint tízszer ennyit, 373 darabot lajstromoz. [F4344] Ennek csak töredék részét is megnézni, kipróbálni, akár csak legfontosabb tulajdonságaikat (vagy azok hiányát) fölmérni reménytelen vállalkozásnak látszik. Gyakorló adatbázisfejlesztő kolléga véleményét idézem egy szakmai levelezési listáról:

„...nincs egy TÉNYLEG jó, mindenre kiterjedő, értelmes és alap logikával megáldott modellező program se. Eddigi legjobbak, amiket használtam, az Oracle-é, és az ErWin, de mindkettőben hatalmas gondok vannak. Az egyik túlzottan szabadjára engedi a usert, a másik meg olyan megkötéseket követel meg amik feleslegesek (mert megesik, hogy az FK mező neve nem azonos a két táblában, mert pl többszörösen alárendelt, vagy csak másodlagos kapcsolat, problémázik ha már

173 L. az „Informatikusként örülünk...” kezdetű bekezdéstől, 75. old.

van alárendelt táblában FK nevű mező. (...) Nincs olyan modellező, ahol szabályokat állíthatsz fel, pl táblarövidítéssel kezdődjenek a saját mezők, domain kezelés fergetegesen rossz, saját szabályrendszer alkalmazás nincs .... Szóval még nincs is mit beleépíteni."[174] [F4306]

Saját magam a tervezés során a fabforce.net DBDesigner 4 for Linux eszközét használtam. Korábban dolgoztam még a Sybase Data Architect nevű eszközével is.

174 L. a CASE-eszközök c. részt, 76. old.

3. Rendszerterv

A rendszerelemzés feladatai

A rendszerelemzésről, annak lényegéről, feladatairól sokan és sokat írtak. Legfontosabb feladatai összefoglalva: a részek meghatározása az elemekig, a hierarchia feltárása; funkció elemzése a részfolyamatokig; alkotók közötti kapcsolatok elemzése, szerkezet feltárása; rendszer és környezete kapcsolata, a működési feltételek feltárása; biztonsági és optimális állapottartomány meghatározása; zavaró hatások előzetes felmérése; a rendszer működéséhez szükséges információs rendszer feladatai; a rendszer működéséhez szükséges belső és külső energiaforrások (pénzügyi is!). [F4289 pp. 162-163.]

„...első lépésünk a rendszer peremének rögzítése. Ez csak látszólag önkényes, hiszen "értelmes perem" csak az lehet, ami valóban a vizsgálat szempontjából összetartozókat fűz össze. (...) A perem egyben megadja a vizsgált rendszer helyzetét is, vagyis elhelyezkedését környezetéhez viszonyítva. Ezután következik a peremen belüli szerkezet feltárása. A rendszer szerkezete: az alkotórészek és a közöttük lévő relációk. (...) Az alkotórészek további bontásával eljutunk az (adott vizsgálati szempont szerinti) elemekig, és ezzel meghatározzuk a vizsgált rendszer hierarchiáját. A rendszer szerkezete még csak a (statikus) vázat adja, a (dinamikus) működés ismeretéhez a kölcsönhatási folyamatokat kell feltárnunk.” [F4289 p. 124.]

Máig fel-felbukkanó hatása van a hajdani ún. IPO-szemléletnek. A betűszó az input-process-output kezdőbetűiből álló rövidítés. Azt a nézetet takarja, amely a számítógépes alkalmazásokat úgy fogja föl, mint a bemeneti adatokat kimeneti adatokká alakító folyamatokat. Ez a szemléletmód az adatbázisok esetében nem alkalmazható, pontosabban káros, mert hibás modellezési szemléletben gyökerezik. A bemenet és a kimenet mindig egy adott felhasználó (felhasználási mód) szempontjából határozható meg, azaz pont a lényeg sikkad el, miszerint az adatbázis közös tulajdon, és egy egységes, globális adatmodellen alapul.[175]

Adatbázisok esetében az ismeretek feldolgozásának van egy optimális menete, amely az adatok közötti viszonyokon és összefüggéseken alapul: az ismeretek logikai rendben és időbeliségükben egymásra

175 L. az Általános és egyedi nézetek c. részt, 31. old.

épülnek. Értelemszerűen előbb (lenne) szükséges pl. a települések és irányítószámaik listáját rögzíteni, és csak utána a hallgatók lakcímeit. Az adatbázis lényegi sajátosságait figyelembe vevő helyes szemlélet szerint az adatbázisban a helyes és teljes *alapadatokat* tároljuk célszerű szerkezetben, s ennek alapján bármilyen szükséges ismeretet az igény felmerülésekor ki lehet elégíteni. Ezt másképpen a „korai bemenet" és a „késleltetett kimenet" kifejezésekkel is illethetjük. [F4260 p. 126.]

Véleményem szerint a szerkezeti modell a döntő, mert bármilyen folyamat csak meglévő szerkezeti elemeken alapulhat, azok között működhet. Természetesen ezen gondolatmenet formálisan megfordítható lehetne, mint hálótervezésnél az eseményközpontú vagy a tevékenységközpontú megközelítés. Esetünkben azonban adatintenzív rendszerről[176] van szó, tehát jogosan tartjuk fontosabbnak az adatszerkezetek modellezését a folyamatokénál.

Ezért tehát a folyamatok modellezése, a DFD-ábra mint a működő rendszer felmérésének és elemzésének eszköze ilyen helyzetben mindenképpen kisebb jelentőségű. Hasznossága azonban megkérdőjelezhetetlen, mert elősegíti a meglévő és működő ügyviteli rend alaposabb és jobb megértését, ami viszont nélkülözhetetlen előfeltétele az eredményes (statikus) adatmodellezésnek is.

Mit? Hogyan? Mivel/kivel?

Információs rendszer fejlesztése „meghatározott elvek, módszerek, eljárások, eszközök olyan tudatos, a rendszer céljának megfelelő alkalmazása, amely a szervezeti és az IT-stratégiával összhangban, az alaptevékenységre és a felhasználó igényeire alapozva:

- a valós probléma felmerülésétől kezdődő folyamattal,
- a feladat megismerési és elemzési munkájának az elvégzése után,
- egy hatékonyabb, számítógéppel támogatott rendszert tervez, és
- Általános és egyedi nézetek valósít meg,

oly módon, hogy a minőségi követelményeket is kielégítő, működőképes, információfeldolgozó-rendszert hoz létre, és felügyeli annak működését." [F4336 p. 285.]

Másképpen a „mit?", a „hogyan?" és a „mivel és kivel?" kérdések együttesen válaszolandók meg, azok megválaszolása egymástól el nem választható. A megvalósítandó cél, annak pontos meghatározása

176 L. az „Adatintenzív rendszerek..." kezdetű bekezdést a 123. oldalon.

befolyásolja a megvalósítás lehetséges eszközeinek és módjainak a körét. Az állítás megfordítása is igaz: adott eszközök és módszerek kiválasztása hatással van a cél elérésére (akár annak lehetetlenségére), a megvalósítás minőségére. Elsődleges azonban az elérendő cél, és a kitűzött cél eléréséhez kell megkeresni a szükséges eszközöket és módszereket, jó esetben a legcélravezetőbbeket, nem pedig fordítva, amint azt az adatmodellezés kapcsán már láttuk.[177]

Fejlesztési elvek

Fő irányelvünk a fogalmi modellezés hangsúlyozása, a fogalmi-logikai-fizikai szintű tervezés következetes végigvitele, egyrészt mert az adatmodellezés kapcsán láttuk, hogy ez az adatmodellezés jellegéből szükségszerűen fakadó követelmény,[178] másrészt pedig mert közben a példa erejével támasztja alá, hogy az SSADM módszer a megfelelő kiegészítésekkel illetve módosításokkal jól alkalmazható lenne a mai körülmények között is. Mivel a jó adatmodell szükséges (de önmagában nem elégséges) előfeltétele bármilyen információs rendszer sikeres fejlesztésének, az előbbi megállapítás kiterjeszthető lenne az SSADM-től különböző fejlesztési módszerekre és eljárásokra is.

Másik igen fontos követelményünk a minél szélesebb körű alkalmazhatóság. Egy tanulmányi rendszer felhasználói köre igen tág, az oktatók, az ügyintézők és a hallgatók mint típusos felhasználók köre tízezres nagyságrendű is lehet. Ilyen körben számos, akár különlegesnek, ritkának tekinthető hardver vagy szoftver fordulhat elő, márpedig olyan megoldást kellene találni, amely az összes lehetséges felhasználó számára biztosítja a rendszer használatának lehetőségét, a lehető legkevesebb előkészület (pl. szoftvertelepítés, -beállítás) mellett. A számítógépek és operációs rendszerek sokféleségéből kiindulva böngészőre és https protokollra alapozott megoldás jön számításba, mert böngésző van (lehet) minden olyan számítógépen, amelyik egyáltalán képes az internetre csatlakozni. Természetesen (?) azért vannak a szabványok, hogy legyen mitől eltérni, tehát figyelembe kell venni a különféle böngészők, sőt -verziók eltéréseit, magyarán: hibáit is. Pontosabban: törekedni kell a böngészőfüggetlen megoldásokra, hogy - szélsőséges esetben - akár szöveges böngészővel (pl. Lynx) is kezelni lehessen a megvalósítandó rendszert.

A böngésző alapú megoldás további haszna, hogy nincs ügyféloldali

177 L. az „A modellezés során...” kezdetű bekezdéstől az Adatmodell c. részben, 24. old.

178 L. 26. old.

egyedi alkalmazás, azt nem kell fejleszteni és karbantartani. Ez tehát a jelen helyzetben egyáltalán elérhető legteljesebb mértékű platformfüggetlenséget jelenti.

A http(s) protokoll nemcsak elősegíti a moduláris felépítés kialakítását, hanem egyenesen kikényszeríti azt. Az egyes modulok mérete arányaiban viszonylag kicsiny, ugyanis bármely típusos tevékenységet egy (esetleg több) modullal lehet lekezelni, elvégezni. Az adatmanipulációs műveleteket az adatbáziskezelő végzi, az egyes modulok lényegében a szükséges sql-utasítások összeállítását és az adatok közvetítését végzik. Így a modulok közötti - nemkívánatos - kölcsönhatások lehetősége csekély, a viszonylagosan kis méret jó hatással van a hibavalószínűségre is.[179]

Hordozhatóság, eszközfüggetlenség

Van legalább két olyan terület, ahol óvatos egyensúlyteremtésre van szükség az egymásnak esetenként ellentmondó lehetőségek között. Az egyik ilyen a hordozhatóság, illetve az eszközfüggetlenség kérdése: alkalmazott megoldásainknak olyanoknak kellene lenniük, amelyek nem kötődnek szorosan egy adott adatbáziskezelőhöz sem.[180] Ezt a célt azonban nem lehetséges elérni. A különböző relációs adatbáziskezelők által használt SQL-nyelvjárások, szintaktikai szabályok ugyanis olyannyira különböznek egymástól, hogy azokat nem lehet közös nevezőre hozni.

Emiatt két dologra lehet és érdemes odafigyelni. Az egyik, hogy törekedni kell a minél kevesebb kezelő-, illetve verziófüggő elem alkalmazására minden olyan esetben, amikor ez egyáltalán lehetséges. Evvel kapcsolatos, hogy a fejlesztés során olyan formai előírásokat kell tenni és betartani, amelyek egy esetleges átállás során megkönnyítik a módosítandó részek azonosítását, lehetővé téve akár a forráskódok gépesített átállítását is.

A másik fontos terület a deklaratív SQL és a procedurális feldolgozások határa, ugyanis nem határvonalról van szó, hanem határterületről. Adott esetben komoly megfontolást igényelhet a határvonal meghúzása, különös tekintettel a különféle kezelők különféle tárolt eljárásaira. Úgy tűnik, hogy nincs olyan megoldás, amely mind az adatbáziskezelő, mind az alkalmazás programnyelve vonatkozásában - egyidejűleg - biztosítani tudná a hordozhatóságot.

Fölmerülhetne még a gazdanyelv megválasztásának szempontja is,

179 L. „A tervezés fontossága...” kezdetű bekezdéstől, 95. old.
180 Vö. a fizikai adatfüggetlenség elvével, 30. old.

de itt a hordozhatóság fogalma nem értelmezett. A kiszolgáló gépen futó programok lehetséges programnyelvei ugyanis elvileg különböznek egymástól, ezek egymással nem csereszabatosak és nem is lehetnek azok. Döntés kérdése, hogy pl. PHP-t használunk-e vagy Perlt vagy - mondjuk - C-t (és melyik megvalósítás melyik verzióját).

Elvárások - szoftverkrízis

A szoftverkrízis fogalma viszonylag régi, a számítástechnika fejlődésének aránylag korai szakaszából származik. 1968-ban, a NATO garmischi konferenciáján hangzott el először. Legfőbb tünetei a szoftverfejlesztési projektek költség- és időkeret túllépése, az alacsony hatékonyság, a gyönge minőség, a követelményeknek való meg nem felelés, a nehezen karbantartható programkód, vagy akár a szoftver el nem készülte. Dijkstra ugyancsak említi a jelenséget a Turing-díj átvétele alkalmából tartott előadásában, 1972-ben. [F4366 pp. 859-866.]

Ezek alapján a legelemibb és legnyilvánvalóbb, de külön említendő elvárás, hogy a rendszer egyáltalán használható legyen, legyen alkalmas a kitűzött célok értelmes módon történő megvalósítására. További általános érvényű elvárás egy optimumfeltétel: a fejlesztésre fordított erőforrások célszerű minimalizálása mellett a kapott használati érték maximuma, tekintettel a szoftverminőséggel szemben szabványokban is rögzített módon támasztott követelményekre.[181]

Követelményrendszer felállítása

A követelmények, elvárások meghatározása során másfél évtizedes felsőoktatásbeli tapasztalataimból indultam ki. Mivel ezt az időt a BMF (és a jogelőd Bánki Donát Műszaki Főiskola) oktatójaként töltöttem, megközelítésem nyilván főiskolánk sajátosságait tükrözi, ennélfogva nem biztos, hogy változtatás nélkül átvihető és alkalmazható más intézmények esetében. A jelenleg hatályos tanulmányi és vizsgaszabályzatunkat vettem alapul. Kérdőíves felmérést készítettem hallgatók körében. Oktató kollégáim, adminisztrációs munkatársaim részletes, ennélfogva időigényes kikérdezését mellőztem, mert az ő idejükkel nem gazdálkodhatok, de ezt - részben - pótolja saját oktatói múltam, illetve az, hogy egy ideig a Neptun kari adminisztrátori feladatait is elláttam. Az oktató és adminisztrátor kollégák célszerűen kiválogatott részhalmazának alapos kikérdezése természetesen nem lenne mellőzhető abban

181 L. 172. old.

az esetben, ha bevezetésre szánt rendszert kellene tervezni és kivitelezni.[182]

A követelményspecifikáció

A követelményspecifikáció során meg kell határozni az új rendszerrel szemben támasztott elvárásokat, rögzíteni azon körülményeket és feltételeket, amelyek a meglévő rendszer megváltoztatását lehetővé ill. szükségessé teszik, továbbá ki kell dolgozni azon új megoldásokat, amelyek hatékonyabban elégítik ki a rendszer alapvető célját. [F4282 p. 540.]

Az elsődleges cél a hatékonyság növelése, ez intézményi szinten az összes emberi, élőerős ráfordítás csökkenését kell eredményezze. Nagyon különleges kivételektől eltekintve egy számítógépes ügyviteli rendszer létjogosultságát pont az élőmunka igényének látványos csökkenése jelenti. Ha ugyanis egy számítógépes rendszerben az ügyek intézése több időt és emberi munkaerőt igényel, mint az annak megfeleltethető papíralapú rendszerben, akkor a papíralapú rendszernél kell megmaradni. Beruházást, fejlesztést akkor érdemes csinálni, ha az valamilyen formában megtérül, valamilyen értelemben véve hasznot hoz.

Ha tehát a hatékonyság, az optimalitás az elsődleges cél, felhasználói és üzemeltetői szempontból egyaránt, akkor érdemes lehet a fejlesztésre többet áldozni, mivel a használat jóval hosszabb ideig tart, mint a fejlesztés. Ezért is szükséges teljes egészében, a problémafölvetéstől az üzemeltetésig, a karbantartással bezárólag egy projektként kezelni a feladatot, és ebben az esetben vizsgálni a tervezési-fejlesztési szakaszra fordított többlet megtérülését, általában: a ráfordítások megtérülését.

A felhasználói hatékonyság egyik fő eleme: a legfontosabb feladatok legkényelmesebb, legkönnyebb (legkevesebb kattintással való) elvégzése.[183] Másik fő elem, hogy a felhasználó számára nyújtott szolgáltatás a lehető legmagasabb szintű legyen, ne követelje meg pl. olyan adatok ismeretét, amelyek az adatbázisban amúgy is benne vannak. Harmadik fő terület, hogy az adatbázis mérete a lehető legkisebb legyen. Ez jelent egy modellezési szempontot: csak az okvetlenül szükséges szolgáltatások nyújtását, továbbá egy logikai tervezési szempontot: a méret és sebesség összehangolására. A hatékonyságot a fizikai szint számos elemével is befolyásolni lehet: törekedni kell arra, hogy az egyes adatelemek a lehető legkisebbek legyenek, minél kevesebb legyen a ki-

182 L. a Fogalomalkotás c. részben (140. old.) a széleskörű egyeztetésről írottakat
183 L. az Ergonómiai szempontok c. részt, 132. old.

használatlan helyfoglalás, illetve az adatok elérését a kritikus igénybevételekre kell optimalizálni.

Adatintenzív rendszerek esetében az adatokon, adatszerkezeteken van a hangsúly. [F4282 p. 541.] A jó (sőt: a lehető legjobb) adatmodell kialakítása kulcsfontosságú. Az adatmodellezés elsődlegessége esetén az „entitások precíz meghatározása a funkcionalitás feltárását megelőzően veszélyes lehet, hiszen így esetleg olyan objektumokkal is foglalkozunk, amelyek vagy a funkcionalitás, vagy pedig az alkalmazás szempontjából lényegtelenek", [F4282 p. 541.] vagy ami a valószínűbb és komolyabb problémákat okoz: esetleg hiányoznak. Ezeket a problémákat azonban jól kezeli az egyed-esemény mátrix SSADM-technika, amely kimutatja a semmilyen tevékenység által nem használt egyedtípusokat és a semmilyen egyedtípusra hatással nem lévő tevékenységeket. Mindkét esetben eldöntendő, hogy az adott egyedtípus vagy tevékenység valóban fölösleges, vagy pedig a tevékenységek, illetve egyedtípusok köre hiányos.

A funkcionalitás vizsgálata során mindenképpen meg kell határozni a kritikus fontosságú tevékenységeket, eseményeket és szereplőket.

A szolgáltatások köre és a hitelesség

Első lépésünk a rendszer és környezetének szétválasztása, a rendszer határainak (peremének) rögzítése.[184] Egy felsőoktatási intézmény elsődleges feladata a hallgatók képzése, és ezen alapfeladatot kell segíteni az informatika eszközeivel.[185] A hatékonyságot meghatározó tényezők egyike a méret.[186] A hatékonyságnak tehát közvetlen előfeltétele, hogy a szolgáltatások köre a a lehető legszűkebb legyen, miközben - természetesen - minden lényeges, az alapfeladat ellátásához szükséges, azt érdemben meghatározó szolgáltatásnak meg kell lennie.

Meg kell tehát határozni az alapfeladat ellátásához okvetlenül szükséges szolgáltatások körét, azaz az oktatással közvetlenül összefüggő feladatok körét. Elsődleges a hallgatók tanulmányi eredményeinek a nyilvántartása, mert ez az alapja minden ügyviteli feladatnak. Ez határozza meg az államvizsgára bocsáthatóságot, az adott félévben fölvehető tárgyak körét, a szakirányválasztást, a félév eredményes vagy eredménytelen lezárását stb.

A hatékonyság egyik előfeltétele, hogy semmilyen fölösleges szol-

184 L. A rendszerelemzés feladatai c. részt, 117. old.
185 L. a „Minden szervezetnek..." kezdetű bekezdést, 13. old.
186 L. az Elvárások - szoftverkrízis c. részt, 121. old.

gáltatást, tevékenységet ne végezzen a rendszer, ugyanakkor minden általa nyújtott szolgáltatás a lehető legmagasabb színvonalú legyen.

A másik alapvető fontosságú kérdés a hitelesség kérdése. Bármely adatbázis létjogosultságát kérdőjelezi meg, ha a benne tárolt adatok hitelességéhez nagyobb kétség férhet, mint amennyit a használati cél és mód alapján még elfogadhatónak tekinthetünk. Esetünkben az eldöntendő kérdés a következő: mi legyen a hiteles dokumentum? Az index, a tanulmányi rendszer adatbázisa, esetleg valami más?

Az adatok hitelességének, pontosabban ellenőrizhetőségének szempontjából fontos, hogy azok legalább két, egymástól független csatornán haladjanak, illetve két, egymástól független helyen és módon tárolódjanak. Történelmileg hagyományos módon ez úgy valósul(t) meg, hogy a megszerzett gyakorlati vagy ún. évközi jegyet a félév végén az oktató beírta a hallgatónál lévő indexbe, aláírta, majd azokról vizsgalapot írt és adott le ugyancsak aláírva a tanulmányi osztályra. Vizsgajegyek esetében ugyanez történt, értelemszerűen vizsgánként külön-külön. A félév végi aláírások esetében az aláírás megtagadását jelezte az oktató a tanulmányi osztálynak, a sikeresen megszerzett aláírást általában a vizsga alkalmából volt szokás beírni az indexbe. Így a tanulmányi osztálynak módja volt ellenőriznie az indexek adattartalmának hitelességét, ami lényeges elem, egyrészt azért, mert az index hiteles okirat, másrészt pedig azért, mert hosszú ideig van az esetlegesen ellenérdekelt hallgató kezében.

A számítógépes nyilvántartás bevezetésével a helyzet alapvetően változatlan maradt: indexbeírás és vizsgalap leadás a korábbihoz hasonló módon történt továbbra is, esetleg és esetenként avval a különbséggel, hogy az oktató az általa a számítógépes nyilvántartásba rögzített eredményeket nyomtatta ki az adatfelvitel után azonnal, írta alá és adta le a tanulmányi osztályra.

A jelenleg használatos számítógépes tanulmányi nyilvántartások (Neptun, ETR) önmagukban nem hitelesek, még de facto sem, de iure pedig még kevésbé. Ez önmagában még nem baj, csak nem szabad figyelmen kívül hagyni, elfelejtkezni erről a körülményről. Éppen ezért aggályosnak találom azt az újabb gyakorlatot, miszerint az oktatók az indexbe nem írják be az eredményeket aláírásukkal ellátva, hanem a vizsgaidőszak lezárása után azokat a tanulmányi osztály kinyomtatja és beragasztja a hallgatók indexébe. Szerintem már maga a beragasztás ténye is problémás - tekintettel az index okirat voltára.

Semmilyen ellenőrzés nincs arra vonatkozóan, hogy a vizsgaidőszak lezárása után kinyomtatott eredmények adattartalma megfelel-e a va-

lóságnak. Pontosabban a lehetséges és szükséges kétirányú ellenőrzésnek csak az egyik fele van meg: a hallgató köteles a nyilvántartásban egy egérkattintással mintegy jóváhagyni a saját osztályzatait, illetve módja van reklamálni, ha nem azt a jegyet látja a nyilvántartásban, amelyet tudomása szerint kapott. A másik irányt viszont senki nem ellenőrzi. Nincs okom azt állítani, hogy a Neptunt ill. az ETR-t a hallgatók tömegesen és rendszeresen manipulálják, vagy akár manipulálhatnák. Aggodalomra ad azonban okot az a körülmény, hogy legalább néhány eset ismertté vált az üzemeltetők számára, amikor illetéktelen manipuláció történt, vagy történhetett volna. Egy ilyen esetnek magam is tanúja voltam, a leleményes hallgató néhány alkalmas kattintás után írási joggal tudta futtatni az alkalmazáskiszolgálón a regedit.exe-t (a Windows rendszerbeállításait közvetlenül olvasó-író eszközt).

Súlyos üzemeltetési hiba az a ténylegesen előfordult eset, ha egy felsőoktatási intézmény a tanulmányi nyilvántartás kiszolgálóján olyan tanúsítványt használ, amelyet a böngészők nem tudnak ellenőrizni. Ezért mind a hallgatók, mind az oktatók megszokják, hogy az ezt jelző hibaüzenetet „tudomásul kell venni”, és nem foglalkoznak vele. Ez azonban megteremti annak lehetőségét, hogy sikeres közbeékelődéses támadást lehessen intézni hallgatói és oktatói jelszavak megszerzésére (pl. az ARP-mérgezés módszerével). [F4385]

Egy számítógépes tanulmányi rendszert lényegileg és formálisan is hitelessé tenni aránytalanul nagy ráfordítást jelentene, nemcsak a fejlesztés, de az üzemeltetés során is. Olyan kiegészítő beruházásokat követelne meg, amelyeket a felsőoktatási intézményektől elvárni nem reális, a hallgatók esetében pedig lehetetlen, enélkül viszont pont az az előny veszne oda, hogy számos ügyet távolról is lehet intézni. Nem elvárható a mai világban (még) a hallgatók tömegeitől, hogy az elektronikus aláírásról szóló 2001. évi XXXV. törvény szerinti digitális aláíráshoz szükséges eszközöket beszerezzék. Ugyancsak nem lenne reális elvárni a felsőoktatási intézményektől azt, hogy legalább kétcsatornás felhasználó-hitelesítést végezzenek egyes bankok gyakorlatához hasonló módon (pl. a bejelentkezés első lépésében megadott felhasználónév-jelszó páros helyessége esetén egy időkorlátos nyolcjegyű szám kiküldése SMS-ben a korábban a felhasználó által megadott mobilszámra, hogy azt a felhasználó a bejelentkezés második lépésében megadja a számítógép-terminálon).

A puszta felhasználónév-jelszó páros megadásán alapuló hitelesítése a felhasználóknak csak abban az esetben elegendő, ha a számítógépes tanulmányi nyilvántartást önmagában nem tekintjük hitelesnek. Az el-

ért eredmények két, független csatornán történő továbbítása (számítógépes rendszer és papíralapú dokumentum) azonban elegendő lehet, a papíralapú dokumentumok hitelességének kikötésével. Az intézmény ugyanis minden hallgatójáról törzslapot köteles vezetni, a hallgatók személyi és tanulmányi adataival.[187] Ugyan a jogszabály nem mondja ki tételesen, hogy ennek papíralapúnak kell lennie, a „törzslap" kifejezés ezt sugallja, a célszerűség szempontjairól nem is beszélve.

Van ennek még egy következménye. Hiteles számítógépes rendszer esetén minden elképzelhető, bár valószínűtlen esemény kezelésére előre fel kell készülni, ami a fejlesztési és tesztelési munkát rendkívül megdrágítja.

Így tehát az a reális megoldás, ha az oktatóknak az elért eredményekről továbbra is le kellene adni a tanulmányi osztályokra a kitöltött és aláírt vizsgalapokat, azt pedig a tanulmányi osztály munkatársai valamilyen módon összevetnék a számítógépes nyilvántartás adattartalmával, majd későbbi ellenőrzés lehetőségének biztosítására irattárba helyeznék. Azaz a hitelesnek tekintett adatokat papíralapú dokumentumok tartalmazzák. Ennek megvan az az előnye, hogy igen komoly üzemzavarok esetén is lehetőséget biztosít a nyilvántartás használatára, ha ez esetben nehézkesen is.

Kurzusra és vizsgára való jelentkezés

Az eddig elért eredmények nyilvántartásán alapul az egyik kulcsfontosságú folyamat kiszolgálása: a tárgyfelvétel, pontosabban a kurzusválasztás minden félév elején. Mivel a hallgatók nem kötelesek - esetenként különféle okok miatt nem tudnak vagy nem akarnak - a mintatanterv szerint haladni, ezért saját maguk dönthetik el, hogy adott félévben mely tárgyakat kívánják tanulni, és hogy a választott tárgyak mely gyakorlatait kívánják látogatni. A kurzusválasztás során minden esetben vizsgálni kell, hogy a hallgató teljesítette-e az adott tárgy előkövetelményét, azaz szerzett-e megfelelő eredményt abból, vagy azokból a tárgyakból, amelyek eredményes elvégzését a kiválasztani kívánt kurzus előfeltételeként meghatározták. Ez elég bonyolult és sokrétű ellenőrzést jelent. Tekintetbe véve a hallgatói létszámot is ez az első és legfontosabb feladat.

A kurzusválasztás jelenti a legnagyobb és legkritikusabb igénybevételt a rendszer számára, mert mindenkori sikerességén az adott félév rendje múlik, és mert a hallgatókat többszörösen is érintheti, tekintetbe

187 2005. évi CXXXIX. törvény a felsőoktatásról, 34. § (2). L. 135. old.

véve a szabad kurzusválasztás és az esetleges órarendi ütközések lehetőségét.

A következő fontos terület ugyancsak az elért eredmények nyilvántartásához kapcsolódik. A tárgyak egy részének a követelménye ugyanis vizsga, azaz a kurzus eredményes teljesítését egy (elégséges vagy annál jobb) vizsgajegy megszerzése jelenti. Ebből következik, hogy a vizsgák nyilvántartása nem választható el az eredmények nyilvántartásától. Ugyanakkor *nem* következik belőle az, hogy a vizsgákra való jelentkezést ne lenne érdemes megvalósítani a rendszeren belül.

Ennek több oka is van. Egyrészt a vizsgára való jelentkezésnek is vannak előfeltételei,[188] amelyeket ellenőrizni kell. Másrészt a papíralapú jelentkezés számos olyan súlyos problémát vetett föl, főként az utolsó időkben, amelyek számítógépes rendszer esetében nem fordulhatnak elő.

A papíralapú jelentkezés úgy történt, hogy minden tanszék kitett a folyosóra egy dossziét, amelyben minden, az adott tanszék által tanított tárgy minden vizsgaidőpontjához tartozott egy vizsgalap, rajta megjelölve egy (piros) csíkkal a létszámkorlát. Voltak hallgatók, akik több vizsgaidőpontra is jelentkeztek mindjárt a vizsgaidőszak elején, elfoglalván a helyet mások elől. Számos alkalommal előfordult, hogy a hallgatók nem vették figyelembe a létszámkorlátot. Sőt, arra is volt példa, hogy a létszámkorlát elérése után a hallgatók egymást húzták ki a jelentkezettek listájából. Mindezen problémák számítógépes megvalósítás esetén nem fordulhatnak elő, mert a szükséges ellenőrzéseket a tárolt adatok alapján gyorsan és biztonsággal el lehet végezni.

Külön kérdés az ún. várólisták kezelése. A várólista azt jelenti, hogy bár az adott vizsgaalkalomra vagy kurzusra a megadott létszámhatárig minden hely foglalt már, de a hallgató feltételesen mégis jelentkezhet, ekkor kerül a várólistára. Ha valaki törli a jelentkezését, akkor a várólista első helyén álló hallgató önműködően bekerül a keretbe. Ennek megítélése vegyes, nálunk többnyire nem alkalmazzák, ugyanis legalább annyi problémát fölvet, mint amennyit megold. A hallgatónak ugyanis időben tudomást kellene szereznie arról, hogy a várólistáról bekerült a tényleges létszámkeretbe. Ez azonban nem garantálható, hiszen vizsgaalkalomról a megelőző nap délig lehet lejelentkezni. A vizsga komolyságát kérdőjelezheti meg, ha a hallgató a vizsgát megelőző nap déli 12-kor értesül arról, hogy másnap reggel mégis vizsgázhat.

Kurzusra való jelentkezés esetén a helyzet hasonló: a regisztrációs

188 L. a „Például egy hallgató…” kezdetű bekezdést, 28. old.

hét legvégén is cserélhet valaki kurzust, az ekkor felszabaduló szabad helyre csak ekkor kerülhetne be a várólistán legelöl lévő hallgató, viszont ha ez mégsem következik be, nincs már sem ideje, sem lehetősége jelentkezéseit átszervezni. Célszerűnek látszik tehát a várólisták mellőzése, amely az adatbázist tovább egyszerűsíti, annak terhelését is csökkentve.

Fölmerül az a kérdés, hogy a vizsgára való jelentkezés kiszolgálása nem ró-e aránytalan többletterhet a rendszerre. A kurzusokra való jelentkezés előzetesen két hétig tart, ezt követi a regisztrációs hét, amikor még lehetőség van a korábbiak javítására. A kurzusokra való jelentkezés tehát jóval rövidebb időtartam alatt zajlik le, mint a vizsgákra való jelentkezés, ugyanis a vizsgaidőszak maga öt hét, de előtte két héttel már lehet jelentkezni a vizsgákra, tehát összesen hét hét áll rendelkezésre. Figyelembe véve azt is, hogy az összes fölvett kurzusnak csak egy része vizsgaköteles, viszont utóvizsga-lehetőség is van, ezért úgy tűnik, hogy a kurzusra való jelentkezés időszaka és tevékenysége valamivel nagyobb terhelést jelent. Mindkét esetben a jelentkezési időszak megnyíltát követő órákban (kb. egy napig) tart a nagy igénybevétel, a csúcsterhelés, utána ez jóval egyenletesebbé válik. Ennek oka az a kézenfekvő körülmény, hogy a jelentkezési időszak legelején mindenki a lehető leghamarabb igyekszik kurzusokat, illetve vizsgaidőpontokat választani, amikor még a legnagyobb az időpontválaszték. Később már csak kisebb javítások, módosítások történnek.

Maga a vizsgára való jelentkezés olyannyira hozzátartozik magához a vizsgához, ezáltal közvetetten az eredmények nyilvántartásához, hogy a rendszer határainak megállapításakor célszerű a rendszeren belülre tervezni, amit előzetes elvárásaink szerint úgy tehetünk meg, hogy az nem jelent olyan többletterhelést, amelyet az amúgy is szükséges teljesítménnyel ne lehetne majd kiszolgálni.

Üzenetküldés és pénzügyek

Az üzenetküldés és a pénzügy területeit érdemes még közelebbről megvizsgálni, ehhez hasonlítsuk össze a hagyományos, papíralapú megoldásokkal.

Papíralapú rendszerben személyre szabott üzenetküldés csak rendkívül kivételes és fontos esetekben történik postai (ajánlott) levél útján. A mindennapi munkakörülmények közötti üzenetküldés általában a hallgatók egészét vagy csoportjait érinti, és hirdetőtáblára való kifüggesztés útján valósul meg. Minden hallgató tudja, hogy a tanulmányi osztály hirdetőtábláját figyelemmel kell (érdemes) kísérni. A technika mai

fejlettségi színvonalán viszont elvárható lehet, hogy ennél közvetlenebb és hatékonyabb megoldást alkalmazzunk. Ezt valósíthatja meg a tanulmányi rendszer üzenetküldő szolgáltatása, amely hallgatók kiválasztott csoportjainak (pl. szakirányválasztás vagy államvizsga előtt álló hallgatóknak üzenne a tanulmányi osztály, adott kurzus hallgatóinak a gyakorlatvezető stb.) juttat el üzenetet elektronikus formában úgy, hogy a hallgatónak a rendszerbe történő következő bejelentkezéskor az üzenet megjelenik, a hallgató annak elolvasását nyugtázza (törli).

A pénzügyek vizsgálatát szűkítsük a hallgatók által fizetendő különféle díjakra, egészen konkrétan a különeljárási díjakra és az ismételt vizsgák díjaira. A hallgatóknak folyósított juttatásoknak ugyanis ügyintézési (adminisztratív) szempontból semmi közük nincs a tanulmányok nyilvántartásához mint alapcélhoz. Egyetlen kapcsolódási pont a tanulmányi ösztöndíj kiszámításának alapjául szolgáló tanulmányi átlageredmény, vagy súlyozott átlageredmény. Ha tehát a (súlyozott) átlagokat és a szükséges személyes adatokat (név, számlaszám stb.) lehet listázni, akkor a továbbiakban a kifizetések lebonyolítása függetleníthető attól a tanulmányi rendszertől, amelyiknek az semmiképpen sem elsődleges feladata.

A hallgatók által befizetendő leggyakoribb tételek: a különeljárási díj és az ismételt vizsga díja. Ezen díjak befizetése feltétele az ismételt vizsga letételének (vagy a más egyéb kötelezettség késedelmes teljesítésének). A hagyományos rendszerben a hallgató kapott egy csekket a tanulmányi osztályon, azt befizette a postán, majd a feladóvevény bemutatása után teljesíthette elmaradt kötelezettségét. Ez természetesen elég időigényes és kényelmetlen eljárás a hallgató szemszögéből, továbbá számottevő mennyiségű munkaerőt köt le a tanulmányi osztályon is.

A számítógépes rendszerre való áttérés után erre a problémára született az ún. gyűjtőszámlás megoldás. Ennek lényege az, hogy a hallgató belátása szerinti, célszerű összeget átutal egy gyűjtőszámlára. A bank értesítése alapján a tanulmányi rendszer adatbázisába bekerül, hogy az adott hallgató pontosan mennyit utalt át, és ennek a keretösszegnek az erejéig a számítógépes rendszerben tud rendelkezni fizetési kötelezettségeinek teljesítéséről tételenként. A felsőoktatási intézmény pedig rendszeres időközönként, vizsgaidőszakban akár naponként összegyűjti ezen rendelkezéseket az adatbázisból, és annak végösszegét egy tételben emeli le a gyűjtőszámláról. Természetesen a hallgató bármikor intézkedhet a saját befizetése fel nem használt részének visszautalásáról saját bankszámlájára.

Ez lényegesen kedvezőbb eljárás a hagyományos, papíralapú megoldásnál, legalábbis mentesíti mind a hallgatókat, mind a tanulmányi osztályt a csekkekért való esetenkénti sorbanállástól, illetve a hallgatókat még külön a postai sorbanállásoktól is. Ennek azonban ára van: bonyolultabb és összetettebb az információs rendszer, fölmerül annak lehetősége, hogy a gyűjtőszámlás megoldás esetleg engedély és jogosultság nélkül végzett pénzintézeti tevékenységnek minősülhet adott esetben, illetve a banki átutalások átfutási ideje plusz az átutalás megtörténtének a tanulmányi rendszerbe történő bekerülési ideje még zökkenőmentes ügymenet esetében is jó pár nap lehet, amely esetenként a hallgató jogos érdekeit is sértheti.

Érdemes tehát megfontolni mind az üzenetküldés mind pedig a pénzügyek intézésének mellőzését, mert ez jelentős mértékű tehermentesítést jelentene egy tanulmányi információs rendszer esetében. Viszont ugyanakkor az is igaz, hogy a mai technikai színvonal mellett jogos az az elvárás, hogy ne kelljen papíralapú hirdetőtáblákat rendszeresen felkeresni, és az is jogos elvárás, hogy ne kelljen se csekkekért sorbaállni, se arra várakozni, hogy mikor kerül be a nyilvántartásba az átutalt összeg.

Mivel a hallgató saját bankszámlaszámát szabadon módosíthatja, fölmerül annak lehetősége, hogy jogosulatlan hozzáférés útján ezen pénzeket valaki illetéktelenül megcsapolja. A vizsgaidőszak, még inkább a tandíjfizetési határidő közeledtével ez (sok) milliós tétel lehet, ha valakinek sikerül a rendszerhez való illetéktelen hozzáférés, vagy hallgatói jelszavak tömeges megszerzése. Pont ennek egy lehetőségére mutattam rá a 2009. évi FIKUSZ konferencián.[189]

Az üzenetküldés lehetőségét azonban meg lehet teremteni anélkül is, hogy a tanulmányi rendszerbe egy teljes vagy akár csak részleges (egyirányú) üzenetkezelést bele kelljen építeni. Mivel elvárható, hogy bármely hallgatónak legyen saját e-mail címe, az üzenetküldést egészen le lehet egyszerűsíteni: a kívánt üzenetet a kiválogatott hallgatóknak közönséges e-mail formájában kell elküldeni, erre a célra pedig a felsőoktatási intézmény amúgy is meglévő saját e-mail kiszolgálóját lehet felhasználni, átadva a célfeladat elvégzését a „célszerszám”-nak. Ez esetben a tanulmányi rendszerre összesen annyi feladat hárul, hogy állítsa elő a címzettek listáját és az üzenet törzsét tartalmazó szöveges állományt az smtp-kiszolgálónak való továbbítás céljából.

Az ellentétes irányú (hallgatótól az oktatónak) üzenetküldés lehető-

189L. a „Súlyos üzemeltetési hiba...” kezdetű bekezdést, 125. old.

ségét pedig még egyszerűbb megteremteni, pontosabban az már adott is - főiskolámon legalábbis így van, és a legtöbb felsőoktatási intézményben úgyszintén -: az intézmény honlapján általában elérhető elektronikus belső telefonkönyv tartalmazza az oktatók e-mail címét is. Tehát nyilvánvalóan fölösleges evvel a tanulmányi rendszert terhelni.

2005-ben egyik egyetemünkön a lehető legrosszabb díjfizetési módot alkalmazták: az eseti csekkekért sorban álltak a hallgatók, majd az ügyintéző azokat egyenként és egyénenként nyomtatta a Neptun segítségével. Ez vizsgaidőszakban másfél munkaerőt kötött le. Itt vezető informatikusként azt a megoldást vezettem be mint gyorssegélyt, hogy készítettem egy, a Neptuntól teljesen függetlenül és http-php alapon működő csekkigénylő programot, amely mindössze annyit csinált, hogy a hallgató neptun-kódját és a felkínált listából kiválasztott befizetési jogcímet jogcímfajtánként összegyűjtötte egy-egy szöveges állományba. Az illetékes ügyintéző ezt kötegelt módban (másolás-beillesztés művelettel) tudta betölteni a Neptunba és kinyomtatni a csekkeket. Ez naponta három alkalommal (reggel, ebédkor és munkaidejének végén) 15-20 percét vette igénybe, az elkészült - névsorban - csekkeket pedig kitette egy erre rendszeresített dobozba a hivatal folyosójára. A korábbi, napi 10-12 óra élőmunka-ráfordítás így lecsökkent maximum 1 órára, a hallgatók által megtakarított sorbanállási időt és bosszúságot nem számítva.

Lehetne azonban még tovább egyszerűsíteni a folyamatot, ez azonban a folyamat érdemi átalakítását követelné meg, ami intézményi döntést igényel. Ezen megoldás lényege az, hogy félév, illetve vizsgaidőszak közben nem foglalkozunk a hallgató fizetési kötelezettségeinek teljesítésével. Fölösleges azt tételenként befizettetni vele és tételenként ellenőrizni. Ellenben a félév eredményes lezárásának újabb, további előfeltétele, hogy minden különeljárási, illetve ismétlővizsga díjat a hallgató egy összegben postán befizetett vagy átutalt. Ez esetben egy alkalommal van szükség legfönnebb egyetlen csekkre (azon ritka esetekben, amikor a hallgatónak nincs saját bankszámlája). A befizetendő összeg viszont pontosan lekérdezhető az adatbázisból az ismételt vizsgajelentkezések darabszáma (illetve a tételenként külön előírt különeljárási díjak listája) alapján. Azaz a hallgató, ha már csekken kívánja befizetni, elvehet a kihelyezett kitöltetlen csekk-kötegből egyet, és kitöltheti a tanulmányi nyilvántartás által közölt végösszeggel. Összesen egyszer kell találkoznia a tanulmányi ügyintézővel, amikor indexét leadja, és ez alkalomból bemutatja a befizetést igazoló szelvényt vagy vagy banki terhelési igazolást.

Ez esetben tagadhatatlanul megvan a kockázata valamekkora veszteségnek, hiszen ha a hallgató számos ismétlő vizsgát tesz adott vizsgaidőszakban, majd utána úgy dönt, hogy az adott intézményben megkezdett tanulmányait egyáltalán nem folytatja tovább, akkor elveszik az általa fizetendő, de nem befizetett díjak összege. Ennek valószínűsége, következésképp összegszerűsége is elég csekély, viszont sokkal kevesebb csekkre van szükség, kevesebb ügyintézői munkára, kevesebb ellenőrzése, és egyszerűbb az adatbázis is.

Egyéb tevékenységek

A kurzusra és a vizsgákra való jelentkezés mint fő szolgáltatások mellett van számos olyan tevékenység, amelyek ezeknek előkészítő és/vagy kiszolgáló tevékenységei, illetve amelyek ezen adatok feldolgozásával azokat hasznosítják.

Az 1-3. táblázatok[190] adatait figyelembe véve adódik néhány érdekes következtetés. A hallgatói órarendek lekérdezése nem tartozik az elsődleges feladatok körébe, nem része az intézményi alapfeladatok ellátásának, illetve az ahhoz közvetlenül és feltétlenül szükséges tevékenységek közé sem tartozik. Tekintetbe véve azonban azt, hogy ez a hallgatók által leggyakrabban használt szolgáltatás, továbbá azt is, hogy a szükséges adatok teljes köre az adatbázisban mindenkor megtalálható, ráadásul az órarend (pillanatnyi állapotának) lekérdezése a kurzusra való jelentkezéshez a regisztrációs időszakban amúgy is szükséges, azon kívül viszont nincs jelentős terhelése a rendszernek, ezt a szolgáltatást nem érdemes elhagyni.

Távlati megfontolás tárgya lehet a meglévő alapadat-bázis felhasználásával egy számonkérő, vizsgáztató modul fejlesztése is. [F4347]

Ergonómiai szempontok

Az ember és munkakörnyezetének tanulmányozása nem öncélú: a munkavégzés biztonsága, a munkavégzők elégedettsége, stresszmentessége közvetlenül befolyásolja a munkavégzés és az ügyintézés hatékonyságát.

A számítógépes munkahelyek kialakítására vonatkozó jogszabályi előírások[191] ismertetése, továbbá a részletes és pontos ergonómiai tervezés nem a jelen dolgozat feladata. A felhasználók számára közvetlenül érzékelhető az ergonómiai megoldások hatása mint pozitív hatás,

190 L. 100. old.
191 50/1999 (XI.3.) Eü.M. rendelet

ha azt nem is tudatosítja. Ezek hiányát viszont szinte mindenki nemcsak észleli, de tudatosítja is.

A meglehetősen sokféle szoftver-ergonómiai szempont között van kettő, amelyik közvetlen kapcsolatban van a kialakítandó információs rendszerrel, sőt annak adatbázisával. Az egyik, hogy az egyes munkatevékenységeket a lehető legkönnyebben lehessen elvégezni, azaz az ésszerűség határain belül a lehető legkevesebb egérkattintással, ami egyben hatékonysági szempont is.[192] A másik ettől nem független: ugyanis az előző követelmény egyik előfeltétele, hogy a felhasználó számára az adott munkafeladat elvégzéséhez szükséges ismeretközlés jól átgondolt legyen, minden szükséges adat legyen ott a képernyőn, amikor azokra szüksége van, ugyanakkor a felhasználót ne terheljük számára érdektelen, technikai adatok közlésével illetve kérésével, mint pl. a tantárgykódok.[193]

Ennek kapcsán két, igen gyakori tevékenységet vizsgálok meg közelebbről, a hallgatók kurzusra és vizsgára való jelentkezését.

Kurzusra való jelentkezésnél a hallgatók elsődlegesen azon tárgyak kurzusai közül válasszanak, amelyekkel a mintatantervhez képest elmaradásban vannak. Ezután a mintatanterv szerint esedékes tárgyak köréből, ezután a mintatanterv szerinti, de jövőben esedékes tárgyak közül, legvégül az intézményen belüli egyéb tárgyak közül. A tárgyfelvétel az adott tárgy adott félévben meghirdetett kurzusainak fajtánként (előadás, gyakorlat, laborfoglalkozás) egyikére történő jelentkezés.

A jelentkezés előzetes, azaz még tart az előző vizsgaidőszak, de már kezdődik a következő félév kurzusaira való jelentkezés, ekkor az előkövetelményeket csak részlegesen lehet ellenőrizni: ha az előkövetelmény valamely más tárgyból való *teljesítés,* akkor a teljesítendő tárgy valamely kurzusába tartoznia kell a hallgatónak, hiszen nyilván lehetetlen olyan tárgyól teljesíteni a követelményt, amelyet a hallgató addig még föl sem vett. A regisztrációs hét kezdetekor (esetleg annak első munkanapjának a végén) elvégezhető a szigorú ellenőrzés: törlendő minden olyan kurzusa a hallgatónak, amelynek előkövetelményét nem teljesítette.

Ha egy tárgy előkövetelménye nem valamely más tárgyból való *teljesítés,* hanem egy másik tárgy egyidejű *felvétele,* (annak valamely kurzusára való jelentkezés), akkor a hallgatónak azon tárgy valamely kurzusát is egyéni döntése alapján ki kell választania, és ez nem tehető önműködővé, mert a hallgató személyes, egyéni döntése határozza

192 L. a Követelményrendszer felállítása c. részt, 121. old.
193 L. a „A tantárgykódokkal kapcsolatos...” kezdetű bekezdéstől, 107. old.

meg, hogy melyik kurzust választja az adott tárgy lehetséges kurzusai közül.

Figyelembe véve, hogy átlagos esetben egy félévben egy hallgatónak körülbelül tíz és húsz közötti számú kurzusa lehet, a kurzusok darabszáma pedig tárgyanként 1 (előadás esetében) és 6 (gyakorlat nagy létszámú évfolyam esetén) között lehet, érdemes egyszerre megjeleníteni az elmaradt tárgyak és az adott félévi tárgyak összes kurzusát, amelyek előkövetelményeit teljesítette vagy teljesítheti, tehát amelyekre egyáltalán jelentkezhet. A mintatanterv esedékes félévén túli, illetve más szak, szakirány tárgyaiból való válogatás nem általános, tehát ezen tárgyak kurzusaira való jelentkezést érdemes a kötelező tárgyaktól jól elkülönítve kezelni.

A vizsgákra való jelentkezéskor hasonló feltételezéssel lehet élni. Egy hallgatónak átlagos esetben 4-6 tárgyból van vizsgája egy vizsgaidőszakban. Érdemes tehát mindazon vizsgaalkalmat egyidejűleg megjeleníteni, amelyekre egyáltalán jelentkezhet a vonatkozó szabályok és körülmények figyelembe vételével, jól láthatóan kiemelve azon vizsgaalkalmakat, melyekre már érvényesen jelentkezett. Ekkor esetleges átjelentkezés esetén nem szükséges megkövetelni, hogy előzetesen és külön lejelentkezzen a szóban forgó alkalomról, az adott tárgyból bármely másik vizsgaalkalomra való jelentkezés egyidejűleg a korábbi jelentkezés törlését is jelenti.

Megvalósíthatósági elemzés

A megvalósíthatóság elemzése röviden felméri, hogy a javasolt információs rendszer vélhetően megfelel-e az intézmény sajátos működési körülményeinek, követelményeinek. Ezen túl azt is felméri, hogy üzletileg indokolt-e a rendszer kifejlesztése.

A javasolt információs rendszer megvalósíthatóságának nemcsak a technikai vonatkozásait kell felmérni és értékelni, hanem azt is, hogy egy ilyen rendszer mennyiben és hogyan segíti az alapcélok elérését.

Az elemzés végén a projektvezetés eldönti, hogy ad-e erőforrásokat a részletes és teljeskörű vizsgálathoz, vagy új irányt keres.

A jogi környezet

Az általános jogszabályi környezet szabályain kívül (mint pl. az 1992. évi LXIII. törvény a személyes adatok védelméről és a közérdekű adatok nyilvánosságáról [F4317]) van néhány specifikusan ide vonatkozó jogszabály, illetve jogszabály-részlet. Ezek:

a 2005. évi CXXXIX. törvény a felsőoktatásról, különösen a 34. § és a 35. §;

a 90/1998. (V. 8.) Korm. rendelet a felsőoktatási tanulmányi pontrendszer (kreditrendszer) bevezetéséről és az intézményi tanulmányi pontrendszerek egységes nyilvántartásáról;

a 79/2006 (IV. 5.) Korm. rendelet a felsőoktatásról szóló 2005. évi CXXXIX. törvény egyes rendelkezéseinek végrehajtásáról.

A 2005. évi CXXXIX. törvény a felsőoktatásról külön részben foglalkozik a felsőoktatási intézményekben használatos, illetve használandó információs rendszerekkel. „Adatkezelés a felsőoktatási intézményekben, a felsőoktatás információs rendszere" cím alatt a 34. § és a 35. §, valamint a 2. sz. melléklet tartozik ide. A legfontosabb részeket idézem:

„34. § (2) A felsőoktatási intézmény nyilvántartja a beiratkozott hallgatókat. A beiratkozott hallgatóról törzslapot állít ki, amelyen feltünteti a hallgató személyes adatait, valamint a tanulmányok folytatásával, a hallgatói jogviszonyból adódó kötelezettségek teljesítésével kapcsolatos adatokat."

A 35. § (1) rendelkezik az országos hatáskörű felsőoktatási információs rendszer (FIR) létrehozásáról, és arról, hogy ennek adattartalma a felsőoktatási intézmények által szolgáltatott adatokon alapul.

„2. számú melléklet a 2005. évi CXXXIX. törvényhez

I. A felsőoktatási intézményekben nyilvántartott és kezelt személyes és különleges adatok

(...)

I/B. A hallgatók adatai

1. E törvény alapján nyilvántartott adatok:

a) felvétellel összefüggő adatok:

aa) a jelentkező neve, neme, születési neve, anyja neve, születési helye és ideje, állampolgársága, állandó lakásának és tartózkodási helyének címe és telefonszáma, nem magyar állampolgár esetén a Magyar Köztársaság területén való tartózkodás jogcíme és a tartózkodásra jogosító okirat - külön törvény szerint a szabad mozgás és tartózkodás jogával rendelkező személyek esetén a tartózkodási jogot igazoló okmány - megnevezése, száma,

ab) az érettségi vizsga adatai,

ac) a középiskola adatai,

ad) a felvételi kérelem elbírálásához szükséges adatok,

ae) a felvételi eljárás adatai;

b) a hallgatói (kollégiumi tagsági, doktorjelölt) jogviszonnyal összefüggő adatok:

ba) a hallgató neve, születési neve, anyja neve, születési helye és ideje, állampolgársága, bejelentett lakóhelyének, tartózkodási helyének címe, értesítési címe és telefonszáma, elektronikus levélcíme, nem magyar állampolgár esetén a Magyar Köztársaság területén való tartózkodás jogcíme és a tartózkodásra jogosító okirat - külön törvény szerint a szabad mozgás és tartózkodás jogával rendelkező személyek esetén a tartózkodási jogot igazoló okmány - megnevezése, száma,

bb) a hallgatói (kollégiumi tagsági, doktorjelölti, vendéghallgatói) jogviszonya keletkezésének és megszűnésének időpontja és módja, a hallgató által folytatott képzés megnevezése, állami támogatottsága és munkarendje, a hallgató tanulmányainak értékelése, vizsgaadatok, megkezdett félévek, igénybe vett támogatási idő, a hallgatói jogviszony szünetelésének ideje,

bc) a külföldi tanulmányok helye, ideje,

bd) az elért és beszámított kreditek, beszámított tanulmányok,

be) a hallgatói juttatások, kollégiumi elhelyezés adatai, a juttatásokra való jogosultság elbírálásához szükséges adatok (szociális helyzet, szülők adatai, tartásra vonatkozó adatok),

bf) a hallgatói munkavégzés adatai,

bg) a hallgatói fegyelmi és kártérítési ügyekkel kapcsolatos adatok,

bh) a fogyatékossággal élőket megillető különleges bánásmód elbírálásához szükséges adatok,

bi) a hallgatói balesetre vonatkozó adatok,

bj) a hallgató diákigazolványának sorszáma, a törzslap azonosító száma,

bk) a hallgató azonosító száma, társadalombiztosítási azonosító jele,

bl) a záróvizsgára (szakmai vizsgára, doktori védésre) vonatkozó adatok,

bm) a hallgatói jogviszonyból adódó jogok és kötelezettségek teljesítéséhez szükséges adatok;

c) a hallgatói pályakövetéssel kapcsolatos adatok;

d) a hallgató adóazonosító jele;

e) az adatokat igazoló okiratok azonosítására szolgáló adatok.

2. A többi adat az érintett hozzájárulásával tartható nyilván.

3. Az adatok továbbíthatók: a fenntartónak valamennyi adat, a fenn-

tartói irányítással összefüggő feladatok ellátásához; a bíróságnak, rendőrségnek, ügyészségnek, a bírósági végrehajtónak, államigazgatási szervnek a konkrét ügy eldöntéséhez szükséges adat; a nemzetbiztonsági szolgálat részére valamennyi adat; a felsőoktatási információs rendszer működéséért felelős szerv részére valamennyi adat; a Diákhitel Központnak a tanulmányok folytatásával összefüggő adatok."

A 79/2006. (IV. 5.) Korm. rendeletből:

„6. § (4) A FIR részére történő elektronikus adatközlés hitelesítése közigazgatási célra alkalmas elektronikus aláírással vagy a Központi Elektronikus Szolgáltató Rendszer üzemeltetőjével kötött megállapodás alapján az Ügyfélkapu által biztosított rendszerben történhet. (...)

7. § (2) A FIR működésével összefüggésben, a Hivatal hatósági feladataihoz kapcsolódóan - saját szolgáltatásaival és a Központi Elektronikus Szolgáltató Rendszer igénybevételével - biztosítani kell

a) a felsőoktatási intézmények számára azokat a követelmény-meghatározásokat (specifikációkat), amelyek alapján a felsőoktatási intézmények fel tudják készíteni az általuk üzemeltetett nyilvántartó rendszereket az adatszolgáltatásra;"

Ezen jogszabályok, illetve jogszabály-részletek egy nagyon fontos dolgot tartalmaznak, és egy másik nagyon fontos dolgot *nem* tartalmaznak.

Tartalmazzák azt, hogy melyek azok a személyes adatok, amelyeket az intézmények kötelesek nyilvántartani a hallgatóikról, másrészt pedig nem tartalmazzák azt, hogy a felsőoktatási intézmények kötelesek lennének valamelyik, már létező információs rendszert használni.

Technikai, emberi és szervezeti környezet

A mai világban a különleges igényű egyedi eseteket leszámítva a hardverellátottság jónak mondható. Jelentős mennyiségű (többnyire személyi) számítógép áll rendelkezésre intézményeinkben, az intézményi szintű szolgáltatásokhoz szükséges nagyobb teljesítményű eszközök is rendelkezésre állnak, és valószínű, hogy ezen a területen vannak legalább ideiglenesen igénybe vehető teljesítménytartalékok is.

A szoftverellátottság kérdése is viszonylag egyszerű. Professzionális, fizetős tervező-, fejlesztő-, kezelőeszközök rendelkezésre állhatnak intézményeink közül legalább néhányban.

Emellett számos olyan szabad szoftver érhető el a világhálón, amelyek mögött igen sokrétű és bőséges alkalmazási tapasztalat áll, azaz használhatóságukat bőségesen bizonyították. Ezek széleskörű mindennapos felhasználásuk okán jelentős támogatói háttérrel rendelkeznek.

Szabad szoftvernek az a szoftver számít, amely teljesíti a következő négy feltételt: a) bármely célra szabadon felhasználható; b) működése szabadon tanulmányozható, sőt módosítható; c) szabadon továbbadható; d) szabadon továbbfejleszthető, a továbbfejlesztés is közreadható. Könnyen belátható, hogy ezen feltételek teljesülése a forráskód hozzáférhetőségét (is) jelenti.

Van egy nagyon fontos további körülmény a szabad szoftverek vonatkozásában. Ez pedig az adatbiztonság kérdése. Üzleti licencelésű, következésképpen zárt forráskódú szoftverek esetében az alkalmazó kezében nincs bizonyíték arra vonatkozóan, hogy a szoftver használata biztonságos (azaz nem végez nemkívánatos tevékenységet), illetve nincs eszköze arra, hogy az általa alkalmazott szoftvert bármikor, bármilyen vonatkozásban megvizsgálhassa egy adott probléma fölmerülésekor. Nemcsak hogy eszköze nincs erre, de az ilyen jellegű kísérletek (visszafejtés) eleve tiltottak.

Felsőoktatásunk informatikai szakembergárdája kiváló, ami Neumann János szülőhazájában talán nem is csoda. Hallgatóink között is vannak még szép számmal, akik komoly szakmai teljesítményekre képesek, erről számos különféle (informatikai) versenyen való részvételünk és az azokon elért helyezéseink tanúskodnak.

A szervezeti háttér adott, amennyiben felsőoktatási intézményeinkről bízvást elmondható, hogy igen közelről és alaposan ismerik a felsőoktatás sajátos adminisztrációs igényeit, szükségleteit. Egy egyetemen vagy egy főiskolán soha nem felejtkeznének el pl. arról az egyébként eléggé ismert körülményről, hogy adott esetben hallgatók felmentést kaphatnak egyes tárgyak teljesítése alól.[194]

Megállapítható tehát, hogy egy ilyen esetleges fejlesztésnek a technikai, emberi, szakmai, jogi és szervezetei feltételei megvannak.

Pénzügyi szempontok

A jelen dolgozat keretei nem teszik lehetővé egy tanulmányi rendszer esetleges fejlesztésének részletes pénzügyi elemzését. Valószínűsíthető azonban néhány megállapítás.

Intézményeink éves szinten tízmilliós nagyságrendben fizetnek ki olyan licencdíjakat, amelyek a jelenleg használt rendszerekkel, azok üzemeltetésével közvetlen kapcsolatban vannak. Egy esetleges fejlesztés pénzügyi létjogosultságát ilyen körülmények között elsősorban az adhatja, ha maga a fejlesztés nem kerül számottevő (azaz piaci áron

194 L. az „Ugyancsak komoly modellezési hiba...” kezdetű bekezdéstől, 107. old.

számított) pénzbe, illetve a majdani üzemeltetés költségei legföljebb ugyanakkorák, min a jelenlegi rendszerek esetén, de inkább alacsonyabbak.

A fejlesztési költségek nagyságrendi csökkentését úgy lehet elérni, ha a fejlesztés szabad szoftverekre alapozottan történik, és annak eredménye is szabad szoftver lesz, a GNU-GPL licenc[195] hatálya alatt. Egy ilyen megoldásnak az előnyei túlmutatnak a tételesen megtakarított fejlesztési költségen (munkadíjon). Dobay Péter az informatikai befektetések kockázatait elemezve rámutat arra, hogy a leggyakrabban kifogásolt területek a befektetések időbeli elhúzódása, valamint az, hogy a végeredményként kapott szoftver nem felel meg az előzetesen megfogalmazott elvárásoknak. Ráadásul az ilyen beruházások sokszor hosszú időperiódus alatt valósulnak meg – a prioritások ez alatt megváltozhatnak. [F4371]

Kérdés, hogy lehet-e ingyenesen, vagy közel ingyenesen érdemi fejlesztőmunkát végezni?

A válasz egyértelmű igen, példa erre a szabad szoftver mozgalom által fejlesztett számos, az egész világon széleskörűen elterjedt és használatos program. Vannak ezek között operációs rendszerek (Linux, FreeBSD), különféle hálózati szolgáltatásokat nyújtó eszközök, mint például webkiszolgáló (Apache), levelezőkiszolgáló (Qmail, Postfix), adatbáziskezelő (MySQL, PostgreSQL), irodai csomagok (OpenOffice.org, KOffice), parancsnyelv-értelmezők és/vagy fordítók (PHP, Python, Perl, gcc)... és a sort lehetne folytatni. Itt szeretném megemlíteni főiskolánk két volt diákját a példák sorában: Farkas János (Chexum) a Linux kernelben a ROM fájlrendszer fejlesztője, Gereöffy Árpád pedig az MPlayer (eredetileg Linuxon, majd számos más operációs rendszeren működő videolejátszó) fejlesztésének elindítója és sokáig vezetője volt.

Természetesen jogosan merül föl az a kérdés, hogy mi késztet egy informatikust arra, hogy ingyen végezzen magas szakmai színvonalú munkát. „Az igazi programozó azzal játszik, amivel dolgozik. Igazából folytonosan csodálkozik, hogy a munkaadója fizet azért, hogy azt csinálja, amit egyébként is tenne”[196]. Komolyra fordítva a szót: a szakmai hírnév, a hivatkozható szakmai teljesítmény, a referencia ilyen késztetések.

Pont a felsőoktatásban lenne meg leginkább a szükséges emberi

195 L. bővebben a http://www.gnu.org címen, nemhivatalos magyar fordítás található a http://www.gnu.hu címen.

196 Az internetes irodalom egyik gyöngyszeme Az igazi programozó. Magyarul megtalálható pl. http://szabilinux.hu/orlando_unix/igazi.html

erőforrás a komoly szakmai munkát végző, TDK-n résztvevő, szakdolgozatot készítő hallgatók és az őket segítő, irányító oktatók körében.

Fogalomalkotás

Az adatmodellezés legelső és igen fontos lépése. Maga a fogalomalkotás is egyfajta modellezés.[197]

A jó adatmodellel szemben támasztott követelmények közül három is közvetlen kapcsolatban van vele, az érthetőség, az egyértelműség és a valósághűség. Ahhoz, hogy ezen követelmények teljesülhessenek, igen gondosan kell eljárni a használni kívánt fogalmak meghatározásakor, figyelemmel arra a körülményre is, hogy bár sok különböző felhasználója lesz az adatbázisnak, de maga az adatmodell mégis egy. A fogalomalkotás jelenti a fogalom megnevezésén túl annak eldöntését, hogy az adott dolog vajon egyed-e avagy valamely egyed egy tulajdonsága (esetleg mindkettő), amennyiben tulajdonság, jelenti az értékkészlet pontos meghatározását is, amely egyébként a majdani adatbevitel ellenőrzésének az alapja. További lehetőség, hogy egy adott jelenség lehet még kapcsolat is más jelenségek között, vagy egy kapcsolatot leíró dolog.

Mivel nem elvárható egy informatikustól, hogy az élet bármely területét olyan alapossággal ismerje, mint annak avatott és gyakorlott művelője, ezért fölmerül az a kérdés, hogy vajon mi garantálja azt, hogy az informatikus fogalomalkotása helyes lesz. Itt két dologra kell figyelnünk.

Az egyik az, hogy a fogalmi rendszer egyben modellalkotó tényező is, tehát nem mindegy, hogy milyen. Matematikai hasonlattal élve fogalmi rendszerünknek bázist kell alkotnia, azaz minden lényeges fogalomnak meg kell lennie, de csak azoknak, illetve ezek egymástól függetlenek kell legyenek, tekintettel a minimalitás követelményére.

A másik körülmény pedig az, hogy az informatikus - mivel többnyire nem gyakorlott és avatott művelője az általa éppen modellezett területnek - alapos és széleskörű eszmecserét kell folytasson a modellezett terület avatott képviselőivel.

A megalkotott fogalmakat - természetesen - dokumentáljuk. Ezen dokumentálást jó esetben a használt CASE-eszköz lehetővé teszi, sőt kikényszeríti, de ha bármilyen okból ez nem megoldott (akár nincs CASE-eszközünk), a megfelelő dokumentálásról magunknak kell gondos-

197 L. 14. old.

kodnunk. [F4334 pp. 112-116.] Ezen dokumentáció felhasználásával lehetőség nyílik arra, hogy kellően széleskörű egyeztetést lehessen végezni az érintettekkel. Az egyeztetés során fölmerülhetnek további szempontok és észrevételek, amelyek módosíthatják az eredeti elgondolásokat.

Ez az egyeztetés - a legszükségesebbeken túl - esetünkben értelemszerűen elmarad, mert nem vagyok abban a helyzetben, hogy számos kollégám idejét ellenszolgáltatás nélkül igénybe vegyem. Ezt valamelyest ellensúlyozza az a körülmény, hogy magam is széleskörű, több mint két évtizedes tapasztalatot szereztem a felsőoktatás működéséről és adminisztrációjáról.

A fogalom érthető és egyértelmű megnevezése, egyed vagy tulajdonság jellege, utóbbi esetben pontos értékkészlete, szükség szerinti (szöveges) leírása alapján tisztázandó az ún. adatszerep, hogy leíró adat-e vagy azonosító, továbbá annak (nemfizikai) ábrázolása. Amennyiben leíró adatról van szó, meghatározandó annak azonosítója is.

Az esetek egy hányadában persze mindez nem okoz komolyabb problémákat, de jelentőségét evvel együtt sem szabad lebecsülni.

A továbbiakban a vizsgajelentkezés terheléses méréséhez szükséges adatmodell-részletben előforduló fogalmakat, illetve adatokat tekintem át vázlatosan.

Hallgatók

Mivel tanulmányi nyilvántartásról van szó, kézenfekvő, hogy legelső helyen a hallgató szerepeljen, azaz mindenképpen szükséges egy „Hallgató” nevű jelenség modellezése. A teljesség igénye nélkül felsorolásszerűen kijelenthető, hogy hasonlóan fontos jelenségek lesznek a „Tárgy”, a „Tanterv”, a „Vizsgaalkalom” is, a későbbiekben még felszínre kerülő további jelenségek mellett.

Mivel vannak saját tulajdonságai, egyedtípussal modellezzük, a továbbiakban fontos tulajdonságait kell megállapítani. Szerencsére ezeket felsorolja a fentebb már idézett jogszabály.[198] Ezek közül kell kiválogatni azokat, amelyeket mindenképpen szerepeltetni kell az adatbázisban.

A válogatás elsődleges szempontja, hogy a kérdéses adatnak van-e (közvetlen) köze személyazonosságához vagy tanulmányaihoz, míg másodlagos szempont, hogy az adott adatra van-e elektronikus adat-

198 I/B. A hallgatók adatai, 135. old.

szolgáltatási kötelezettsége az intézménynek, vagy van-e arra viszonylag gyakran szüksége saját magának.

Így tehát az adatmodellből kihagyom pl. az idegen állampolgárság esetén a Magyar Köztársaság területén való tartózkodás jogcímével és az arra jogosító okmánnyal kapcsolatos adatokat.

Elsősorban szükségesek tehát a hallgató személyi adatai: neve, születési neve, anyja neve, születési helye és ideje, állampolgársága, bejelentett lakóhelyének, tartózkodási helyének címe, értesítési címe és telefonszáma, elektronikus levélcíme.

A hallgató tanulmányi előéletének adataiként középiskolájának és érettségi vizsgájának adatai és a felvételi eljárás vonatkozó adatai szerepelnek.

A hallgató felsőoktatási tanulmányaival kapcsolatos elsődleges adatok: a hallgatói jogviszony keletkezésének és megszűnésének időpontja és módja, a hallgató által folytatott képzés megnevezése, állami támogatottsága és munkarendje, a hallgató tanulmányainak értékelése, vizsgaadatok, megkezdett félévek, igénybe vett támogatási idő, a hallgatói jogviszony szünetelésének ideje, a megszerzett és beszámított kreditek.

A hallgató más rendszerekbeli azonosító adatai: törzslapjának azonosító száma, társadalombiztosítási azonosító jele, adóazonosító jele, bankszámla száma és diákigazolványának száma.

A hallgató egyéb adatai, mint pl. a különféle juttatásokkal kapcsolatos (szociális) adatok, fegyelmi és kártérítési ügyek stb., ezek pontos megállapítása mindenképpen alapos és részletes konzultációt igényel arra illetékes tanulmányi osztályi munkatársakkal.

A hallgatók azonosítására alkalmas adatok történelmi hagyomány szerint a neve, az édesanyja neve, születési helye és ideje. Ezek együttes használata azonban a legtöbb esetben körülményes, ennélfogva a hallgatók azonosítására egy mesterséges azonosítót (sorszámot) célszerű használni.

A felsorolt jellemzők között nincs olyan, amelyiknek esetleges időbeli változásait nyomon kellene követni, adatváltozás esetén a szóban forgó adatot a szükséges feltételek fennállása esetén javítani kell.

Főiskolánk esetében az aktív hallgatók száma tízezer körül mozog.

Tantárgyak

A tantárgyakkal kapcsolatos legfontosabb ismérvek: a tárgy pontos neve, mely szakokon és tagozatokon, mely tanterv részeként oktatják, melyik tanszék oktatja, a tantárgyfelelős oktató személye, előkövetel-

ménye (tárgy vagy tárgyak, eredmény vagy egyidejű felvétel), heti előadási, gyakorlati és laboratóriumi óraszámai, a tárgy teljesítésének elismerése (évközi jegy, vizsga, teljesítés), oktatási cél, tematika, tananyag ütemezése, félévközi követelmények (zh, feladat stb.) és azok határideje, kötelező és ajánlott szakirodalom.

A tantárggyal kapcsolatos nehézségek ott kezdődnek, hogy már önmagával való azonosságát sem könnyű meghatározni. Azonos nevű tantárgyakat különböző képzéseken különböző tartalommal és követelményrendszerrel is lehet oktatni, ezek azonban nem tekinthetők azonosnak egymással, éppen eltérő tartalmuk és követelményeik okán. Tovább árnyalja a képet, hogy ugyanazon tartalmú és követelményrendszerű tárgyat is lehet különböző képzéseken oktatni, ez esetben nyilván nem tekinthetők különbözőknek csak azért, mert két külön szakon is oktatják. Triviális példa erre a testnevelés.

További nehézség forrása, hogy egy tantárgy tartalma ugyanazon képzésben való oktatása során is változik apránként az idők folyamán.

A tantárgynak tehát nincs természetes azonosítója, önmagával való azonosságát egy mesterséges azonosító (pl. sorszám) azonos értéke jelenti. Azon jellemzői, amelyek idők folyamán megváltozhatnak, idősorosan modellezendők, azaz szükséges nyilvántartani az érvényesség kezdetét és végét (esetünkben tipikusan félévét), úgy, hogy minden korábbi állapot visszakereshető legyen. Reális igény lehet ugyanis az pl., hogy egy volt hallgatónk számára kell igazolást kiállítani az általa korábban teljesített tárgy tartalmáról, egy másik intézménybeli kreditbeszámítási kérelemhez.

A tantárgyak különféle képzések tanterveihez fűződő kapcsolata kölcsönösen, mindkét irányban többszörös (N:M).

Kurzusok

A kurzusok esete valamelyest egyszerűbb a tárgyakénál. Egy adott tantárgynak több kurzusa is lehet, akár egyazon félévben is. Leíró tulajdonságai a féléve (pl. 2008/09 II.), típusa (előadás, gyakorlat, labor), lehet több helyszíne (pl. N.2.224 terem) és időpontja, pontosabban napja, kezdete és vége (pl. hétfőn a 3-4. órában). További leíró tulajdonsága az oktató, vagy oktatók neve.

Kurzust többféleképpen kell tudni meghirdetni. Szükséges, hogy lehessen egy adott kurzusra több különböző képzés hallgatóinak is jelentkeznie, de az is szükséges lehet, hogy a különböző képzések hallgatóinak különböző kurzusokat lehessen meghirdetni, a helyi sajátosságok,

pl. telephelyek száma és azok egymástól való távolságának függvényében.

Mindenképpen a kurzus szintjén kell arról rendelkezni, hogy az adott kurzus melyik képzés hallgatóinak a számára van meghirdetve, és semmiképpen sem a tárgyak szintjén.[199] Az ellentmondástól való mentességet külön ellenőrzés biztosíthatja: egy tárgy bármely kurzusát csak azon tantervek (képzések) hallgatói számára lehet meghirdetni, amely tantervek azt tartalmazzák. Ezen kívüli hallgatók az általános szabályok szerint (mint bármilyen más, tantervükben nem szereplő tárgy kurzusára) jelentkezhetnek. A kurzusok és a tantervek kapcsolata tehát kölcsönösen többszörös (N:M).

Fontos modellezési kérdés, illetve döntés, hogy a kurzus a tárgyhoz kapcsolódik-e (a tárgynak van kurzusa), vagy pedig a tantervek és a tárgyak kapcsoló egyedtípusához. Ha ugyanis az utóbbihoz, akkor az egyben szerkezeti korlátja a felvehetőségnek, és csak az adott tanterv szerinti képzés hallgatói jelentkezhetnek az adott kurzusra. Ez viszont ellentmond a fentebb megfogalmazott követelménynek, miszerint különböző képzések hallgatói számára is kell tudni közös kurzust meghirdetni. Ezért tehát a kurzus a tárgyhoz kell kapcsolódjon. A felvehetőséget ennélfogva a kurzusok és tantervek N:M kapcsolatát felbontó kapcsoló egyedtípus írja le, és külön érvényesítési korlátként kell előírni, hogy csak azon tantervek hallgatói számára lehetnek az egyes kurzusok felvehetők, amely tanterveknek részei az adott kurzusok tantárgyai.

Tanterv

A tanterv, pontosabban mintatanterv fogalma könnyen összetéveszthető a képzésével. Az érdemi különbség közöttük az, hogy egy adott képzés (pl. szervező informatikus) idők folyamán, sőt akár egyidőben is több különböző tanterv szerint folyhat. Egyrészt a különböző képzési formák (vagy tagozatok: pl. nappali, levelező), másrészt pedig az időnként szükségszerűen fölmerülő tantervi változások miatt.

A tanterv tartalmazza egy adott képzés (szak) elvégzéséhez szükséges összes tárgyat, ezek legelőnyösebb elosztását az egyes félévek között. A tanterv-táblázatok ezen túl tartalmazzák azt, hogy a tárgyak hány kreditpontot érnek, mennyi a heti óraszámuk kurzusfajtánként (előadás, gyakorlat), hogyan történik a tárgy teljesítésének elismerése (pl. vizsga), és mi a tárgy előkövetelménye. Ez utóbbiak azonban nem

199 L. a „A kezdeti időkben problémát...” kezdetű bekezdést, 107. old.

a tanterv tulajdonságai, hanem az adott tantárgy tulajdonságai, ugyanis semmiképpen sem lehet azonosnak tekinteni két olyan tárgyat, amelyeknek a neve bár azonos, de óraszámuk pl. különböző.

A tantervek különféle blokkokra oszlanak, amelyek a tárgyak célszerű csoportosítását teszik lehetővé (pl. általános alapozó tárgyak, szakmai alapozó tárgyak). Mivel a záróvizsgára bocsátás feltétele a megfelelő mennyiségű kreditpont összegyűjtése mellett a tantervben előírt valamennyi tárgy eredményes teljesítése azok csoportosításától függetlenül, ezért ezen csoportosítások modellezése feltehetően elkerülhető.

A tantervek kapcsolata a tantárgyakkal (tartalmazás) és a kurzusokkal (felvehetőség) kölcsönösen többszörös (N:M). Szükséges tehát egy-egy kapcsoló egyedtípus szerepeltetése.

Vizsgaalkalom

Fontos jellemzői - értelemszerűen - a helyszíne, időpontja (kezdete, vége), fajtája („közönséges" vizsga, évközi jegyet vagy aláírást pótló vizsga) és a vizsgára jelentkezők létszámkorlátja. Némileg bonyolítja a helyzetet az oktatói vizsgakombinálás jogos igénye. Előnyös volna, ha lehetne meghirdetni különböző vizsgákat azonos időpontra és helyszínre egyesített létszámkorláttal. Utóbbi a leglényegesebb követelmény, hiszen az azonos időpontra és teremre vonatkozó ellenőrzést korábban is meg lehetett kerülni olymódon, hogy az egyik vizsgát az oktató kiírta az „igazi" vizsgaterembe, a másikat a saját szobájába, majd az ajtajára kitett egy papírt, hogy a vizsga fizikai helyszíne az „igazi" terem.

Ez az elvárás nemcsak egy oktató két különböző vizsgája kapcsán merül föl, hanem két különböző oktató két külön vizsgája kapcsán is. Írásbeli vizsgát véve alapul a kézenfekvő fölső létszámkorlát ugyanis a terem befogadóképessége.

Fölmerül az a kérdés is, hogy mihez tartozik, mihez kapcsolódik a vizsgaalkalom egyedtípus? A kurzushoz kapcsolni nem szerencsés, mert egy adott tárgynak számos - a vizsgák szempontjából egyenértékű - kurzusa lehet. A tárgyhoz kapcsolni is problémás, mert ez esetben nincs mód az esetlegesen több képzés számára is elérhető tárgy vizsgaalkalmait egymástól elkülöníteni még akkor sem, ha ez valamilyen okból szükséges lenne. Ezért tehát a vizsgaalkalmakat a tantervek és tantárgyak kapcsoló egyedtípusához kell kapcsolni, és ezen kapcsolat ugyancsak mindkét irányból többszörös (M:N), azaz felbontandó.

Elvárás tehát, hogy korlátozás nélkül lehessen azonos helyszínre és

időpontra több vizsgát is meghirdetni, de a véletlen ütközések elkerülése végett ilyen esetekben valamilyen figyelmeztetés beépítése szükséges.

Jelentkezés

A vizsgákra történő jelentkezés az egyik sarkalatos pontja a tanulmányi rendszereknek, amely látványos előnyöket kínál a papíralapú jelentkezéshez képest.[200] A vizsgára való jelentkezés számos ellenőrzendő feltételét fentebb már leírtam,[201] az ezen feltételek ellenőrzéséhez szükséges elemeket mindenképpen szerepeltetni kell a modellben.

A jelentkezés a hallgatók és a vizsgaalkalmak közötti, felbontandó M:N kapcsolat.

A jelentkezés legalapvetőbb előfeltétele, az aláírás megléte azonban néhány problémát vet föl, amelyeket nem szabad figyelmen kívül hagyni. Tanulmányi és vizsgaszabályzatunk szerint a vizsgára bocsátás előfeltétele az aláírás megszerzése. A szorgalmi időszak utolsó napján legkésőbb az oktatónak könyvelnie kell a megadott aláírásokat, a megtagadott, de pótolható aláírásokat, illetve a letiltásokat. Letiltás esetén a hallgató az adott tárgyból nem vizsgázhat, az aláírás pótlólagos megszerzésére sincs lehetősége. Megtagadott aláírás esetén a vizsgaidőszak elején egy alkalommal kísérletet tehet a hallgató az aláírás megszerzésére ismétlővizsga jelleggel, egy külön erre a célra kiírt vizsgaidőpontban.

A fenti, elméleti helyzet a gyakorlatban kétféle problémát vet föl. Az egyik egy emberi probléma: nem feltétlenül sikerül az oktatónak a szorgalmi időszak utolsó napján elvégeznie teljeskörűen az aláírásokkal kapcsolatos adminisztrációt. Esetleg azért, mert pont azon a napon délután írat még pótzárthelyit nagyobb létszámú hallgatósággal.

A másik probléma, hogy a vizsgákra történő jelentkezés hetekkel korábban megkezdődik, azaz a hallgató úgy jelentkezhet bármely vizsgára, hogy az aláírása nincs, a legtöbb esetben még nem is lehet lekönyvelve.

Éppen ezért a vizsgára való jelentkezést a kurzusokra való (előzetes) jelentkezéshez hasonlóan kell megoldani. Az általános jelentkezési szabályokat avval kell kiegészíteni, hogy - természetesen - továbbra sem jelentkezhet az, akinek az adott tárgyból már be van írva a letiltás. Akinek ilyen nincs, annak a jelentkezése előzetesnek számít. Mivel vizsgára jelentkezni a vizsga előtti napon délig lehet, ezért - mondjuk - a

200 L. az „Ennek több oka...” kezdetű bekezdéstől, 127. old.
201 L. a „Például egy hallgató...” kezdetű bekezdést, 28. old.

megelőző napon reggel le kell futtatni egy ellenőrzést, amely az adott vizsgára jelentkezettek közül törli azon hallgatókat, akiknek még mindig nincs beírva az aláírásuk. Ez a szigorú forgatókönyv. Ennek megengedő változata szerint csak azon hallgatókat kell törölni a jelentkezettek sorából, akiknek a „Megtagadva" vagy a „Letiltva" bejegyzést beírták, akinek semmilyen bejegyzése nincs, az marad. (Természetesen vizsgajegy beírását addig nem szabad elfogadnia a rendszernek, amíg az aláírás nincs lekönyvelve.) Kiegészítő ellenőrzésként letiltás esetén az adott tárgyból az összes vizsgaidőpont ellenőrzendő, és a hallgató esetleges jelentkezése törlendő.

Kapcsolók

A hallgatók és kurzusok közötti FELVETTE egyedtípus, a kurzusok és a tanterv közötti FELVEHETŐ, a tárgyak és a tantervek közötti TÁRGY_TANTERVBEN, valamint a vizsgaalkalmak és a „Tárgy_tantervben"-ek közötti VIZSGÁJA_VAN, továbbá a hallgatók és a vizsgaalkalmak közötti VIZSGÁRA_JELENTKEZÉS egyedtípusok mint kapcsolók szerepelnek az N:M kapcsolatok felbontása miatt. Ezen kapcsolóknak az adott kapcsolatra jellemző saját tulajdonságaik lehetnének, mint pl. a FELVETTE egyedtípusnak szükség esetén lehetne tulajdonságtípusa pl. a jelentkezés időpontja, vagy akár az, hogy a jelentkezés milyen IP-címről történt.

Eredmények

Bár a vizsgákra történő jelentkezéshez nincs közvetlen köze az elért eredményeknek, mégis érdemes kicsit közelebbről megvizsgálni, mert jól példázza, hogy az adatmodellezés során érdemes tovább gondolkodni az első ötletnél. A vizsgák eredményeit első megközelítésben hajlamosak lennénk a hallgató és a vizsgaalkalom közötti kapcsoló egyedtípushoz (JELENTKEZÉS) társítani, hiszen a vizsgajegy a vizsgán születik. Ez azonban nem a legjobb megközelítés, az eredmények későbbi kezelésének végiggondolása világosan rávilágít erre. Jobb megoldás, ha a vizsgajegyeket a más típusú eredményekkel (pl. évközi jegy) együtt, egységesen kezeljük. Ennek felismerése után viszont adódik, hogy jobb helye van az elért eredményeknek a kurzusok és a hallgatók kapcsolójánál, a FELVETTE egyedtípusban. Mivel egy kurzusfelvételhez több eredmény is tartozhat, ezért az Eredmény egyedtípus 1:N alárendeltje annak. Tulajdonságtípusai lehetnek: a teljesítés ténye, a teljesítés tényét mikor és ki rögzítette, az elért eredmény fajtája és értéke, az eredményt mikor és ki rögzítette. Az eredmény fajtája azt határozza meg,

hogy az értékelés 1-5 közötti jegyekkel történik-e, vagy (jól) megfelelt és nem megfelelt minősítéssel, illetve az aláírva, megtagadva, letiltva minősítések valamelyikével, az érték pedig az eredményfajtának megfelelő egyik érvényes érték.

Fogalmi szintű adatmodellezés

A fogalmi szintű adatmodell a majdani adatbázis alapja, mintája. Tartalmazza az adott cél szempontjából fontos jelenségek, azok tulajdonságainak és a közöttük lévő fontosnak ítélt jelenségviszonyokat. Az adatmodell meghatározza a kimeneti lehetőségeket is, hiszen csak olyan nézet vagy lista állítható elő, amely az adott szerkezet alapján, abból levezethető. (Ezért is fontos a lekérdezési utak meghatározásának SSADM-technikája.[202]) Az adatmodell szerkezetéből fakadó szerkezeti korlátok és az előírandó érvényesítési korlátok[203] együttesen pedig a bemenet lehetőségeit határozzák meg jelentős mértékben, ha nem is a teljesség igényével, hiszen a gépi ellenőrzés a legtöbb esetben nem lehet teljeskörű, épp emiatt van szükség az alkalmazási adatszabványokra.[204]

Mivel az ember vizuális lény, szerencsés az adatmodell szerkezeti kialakítását valamilyen elfogadott módon ábrázolni. Charles Bachman már 1969-ben foglalkozott az adatszerkezetek ábrázolásának kérdésével. [F4325] Azóta számos ábrázolási mód és azok számos változata alakult ki, azonban lényegük közös: valamilyen módon képszerűen láthatóvá tenni az adatmodell szerkezetét, ezen belül is jelezni a kapcsolatok legalapvetőbb jellemzőit: azok számosságát és kötelezőségét.

Logikai szintű tervezés

A hatékonyság az egyik fő szempontja a logikai szintű tervezésnek[205]. Esetünkben ez az elsődleges fontosságú szempont. Csúcsterhelésre kell méretezni a rendszert, mert a vizsgákra és a kurzusokra való jelentkezések időszakában szükségszerűen rendkívül nagyszámú kérést kell kiszolgálni viszonylag rövid idő alatt.

A hozzáférési szempontokat természetesen nem lehet figyelmen kívül hagyni. A hozzáférés szabályozása azonban általában nem olyan

202 L. 90. old.
203 L. „Az adatmodell rendszer is…” kezdetű bekezdést, 24. old.
204 L. 156. old.
205 L. 28. old.

erőforrásigényes, hogy érdemben rontaná a hatékonyságot. Fő lehetőségeink: tábla, oszlop, sor szinten kezelni a jogosultságokat, ha az alkalmazott adatbáziskezelő rendszer képes erre. A másik alapvető lehetőség, hogy az egyes műveleteket végző programmodulok ellenőrzik a felhasználó jogosultsági szintjét, hogy az adott műveletet jogosult-e elvégezni.

Ehhez az szükséges, hogy a rendszer felhasználóinak az egyes műveleteket végző programmodulokra (funkciók) és/vagy azok csoportjaira vonatkozó jogait (futtathatja avagy sem) tároljuk a technikai adatok között, így ezt is tervezni kell. A legegyszerűbb esetben ez a felhasználók és a programmodulok N:M kapcsolatát jelent, amelyet N:M mivoltánál fogva természetesen föl kell bontanunk, a kapcsoló egyedtípus előfordulásai jelentik, hogy az adott felhasználó az adott modult futtathatja, az alapértelmezett tiltás kivételeként. Többrétű lehetőséget biztosít, ha a kapcsoló egyedtípusnak a jogosultság jellegét és mértékét leíró egy vagy több tulajdonságtípusa is van.

Ezt a körülményt a fogalmi modellezés keretei között nem vizsgáltuk, nem is vizsgálhattuk, hiszen annak illetékességi körén kívül esik, mivel a kiválasztandó eszköz egyik tulajdonságától függ.

A technikai adatok (mint pl. a jogosultságokra vonatkozóak) mindenképpen a lehető legteljesebb mértékben elkülönítendők az eredeti cél szerinti adatoktól, jellegük, szerepük, kezelésük célja ugyanis egészen más, összekeverésük csak problémákat okozhat a későbbiekben. Egyes adatok besorolása azonban nem szükségképpen egyértelmű, az gondos modellezési megfontolások eredményeképp dönthető el.

Az adatvédelem szempontja nem elválasztható a hozzáférés szempontjaitól. A rendszer üzemeltetői, akik az üzemeltetés kapcsán szükségszerűen hozzáférhetnek, esetenként hozzá is férnek mások (mindenki) összes adatához, titoktartási nyilatkozatot tesznek. Nincs olyan technikai megoldás (PC-architektúrán legalábbis nincs), amely teljeskörűen biztosítaná, hogy az üzemeltetés különféle feladatai során az üzemeltetők maguk se ismerhessék meg az adatokat.

A hatékonyság növelésére szükség esetén van további lehetőség. Például elképzelhető egy olyan megoldás a terhelési csúcs csökkentésére, amelynek során a beérkező hallgatói jelentkezéseket mint puszta óhajokat egy ÓHAJ táblában rögzítjük minden ellenőrzés nélkül. Ez az adott egy darab táblára korlátozódó 'insert' utasítások végrehajtását jelenti, hallgatónként és tárgyanként egyet (esetleg többet, ha a hallgató figyelmetlen vagy csalni próbál vagy téved).

Mivel az SQL adatbeszúrási utasításai atomiak, ezek problémamen-

tesen és a lehető leggyorsabban végrehajtódnak. Ha az adatbázis még ezt sem győzi a kívánt ütemben, akkor sajnos más megoldást kell keresni, mert feloldhatatlan ellentmondások vannak a fölmerülő igények és a rendelkezésre álló erőforrások között. Ennek az ÓHAJ táblának a feldolgozása pedig olyan ütemben történik meg, ahogy azt az adatbázis és környezete győzi: egy (pontosabban egyetlen) külön program szép sorjában veszi az itt tárolt hallgatói óhajokat (valahány darabonként), elvégzi a vizsgajelentkezés kapcsán szükséges ellenőrzéseket, ennek sikeressége esetén a tényleges jelentkezést rögzíti (az ellenőrzés sikertelensége esetén nincs tennivaló, az óhajt figyelmen kívül hagyja), majd a feldolgozott sort törli az ÓHAJ táblából.

Fizikai szintű tervezés

A különféle technológiák fontosságát és szerepét korábban már vizsgáltam, ezen vizsgálataim eredményét a 2005. évi, III. MEB konferencián ismertettem. [F4369]

A fizikai tervezés során semmilyen, az alkalmazott szoftverekre vagy hardverre (MySQL, PHP, Apache) specifikusan jellemző sebességfokozó megoldást, lehetőséget nem használtam ki. Mivel alapgondolatom a minőségi fogalmi szintű modellezés elsődlegessége, az alkalmazott specifikus eszközök teljesítményfokozó lehetőségeinek (pl. táblaszegmentálás, klaszterezés) alkalmazása torzítaná ennek alátámasztását, nehezítené az elvégzendő mérés értékelését.[206]

A méréshez közvetlenül használt, fizikai szintű adatbázisdefiníció az alábbi:

```
-- MySQL dump 10.10
--
-- Host: localhost    Database: phd
-- ------------------------------------------------------
-- Server version    5.0.24a

/*!40101 SET
@OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET
@OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CON-
NECTION */;
```

206 L. a Mérési terv c. részt, 175. old.

```
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS,
UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHE-
CKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `ALAPERTELMEK`
--

DROP TABLE IF EXISTS `ALAPERTELMEK`;
CREATE TABLE `ALAPERTELMEK` (
  `AlapertelmekAzon` bigint(20) unsigned NOT NULL
auto_increment,
  PRIMARY KEY  (`AlapertelmekAzon`),
  UNIQUE KEY `AlapertelmekAzon` (`AlapertelmekAzon`)
) ENGINE=InnoDB DEFAULT CHARSET=latin2;


--
-- Table structure for table `FELEVEK`
--

DROP TABLE IF EXISTS `FELEVEK`;
CREATE TABLE `FELEVEK` (
  `FelevNeve` char(12) NOT NULL,
  `RegHetKezdete` date default NULL,
  `SzorgIdKezdete` date default NULL,
  `SzorgIdVege` date default NULL,
  `VizsgaIdKezdete` date default NULL,
  `VizsgaIdVege` date default NULL,
  PRIMARY KEY  (`FelevNeve`)
) ENGINE=InnoDB DEFAULT CHARSET=latin2;
```

```
--
-- Table structure for table `FELVETTE`
--

DROP TABLE IF EXISTS `FELVETTE`;
CREATE TABLE `FELVETTE` (
  `HallgatoAzon` bigint(20) unsigned NOT NULL,
  `KurzusAzon` bigint(20) unsigned NOT NULL,
  PRIMARY KEY  (`HallgatoAzon`,`KurzusAzon`),
  KEY `KurzusAzon` (`KurzusAzon`),
  CONSTRAINT `FELVETTE_ibfk_1` FOREIGN KEY (`HallgatoA-
zon`) REFERENCES `HALLGATO` (`HallgatoAzon`) ON DELETE
CASCADE,
  CONSTRAINT `FELVETTE_ibfk_2` FOREIGN KEY (`KurzusAzon`)
REFERENCES `KURZUS` (`KurzusAzon`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin2;


--
-- Table structure for table `HALLGATO`
--

DROP TABLE IF EXISTS `HALLGATO`;
CREATE TABLE `HALLGATO` (
  `HallgatoAzon` bigint(20) unsigned NOT NULL auto_incre-
ment,
  `Neve` char(32) default NULL,
  `SzuletesiHelye` char(4) default NULL,
  `SzuletesiIdeje` date default NULL,
  `AnyjaNeve` char(32) default NULL,
  PRIMARY KEY  (`HallgatoAzon`),
  UNIQUE KEY `HallgatoAzon` (`HallgatoAzon`)
) ENGINE=InnoDB DEFAULT CHARSET=latin2;


--
-- Table structure for table `JELENTKEZES`
--

DROP TABLE IF EXISTS `JELENTKEZES`;
```

```
CREATE TABLE `JELENTKEZES` (
  `HallgatoAzon` bigint(20) unsigned NOT NULL,
  `VizsgaAlkalomAzon` bigint(20) unsigned NOT NULL,
  PRIMARY KEY  (`HallgatoAzon`,`VizsgaAlkalomAzon`),
  KEY `VizsgaAlkalomAzon` (`VizsgaAlkalomAzon`),
  CONSTRAINT `JELENTKEZES_ibfk_1` FOREIGN KEY (`Hallgato-
Azon`) REFERENCES `HALLGATO` (`HallgatoAzon`) ON DELETE
CASCADE,
  CONSTRAINT `JELENTKEZES_ibfk_2` FOREIGN KEY (`VizsgaAl-
kalomAzon`) REFERENCES `VIZSGAALKALOM` (`VizsgaAlkalom-
Azon`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin2;


--
-- Table structure for table `KURZUS`
--

DROP TABLE IF EXISTS `KURZUS`;
CREATE TABLE `KURZUS` (
  `KurzusAzon` bigint(20) unsigned NOT NULL auto_incre-
ment,
  `Fajta` char(2) default NULL,
  `Hely` char(8) default NULL,
  `Napja` char(3) default NULL,
  `Paritas` char(2) default NULL,
  `Kezdete` tinyint(4) default NULL,
  `Vege` tinyint(4) default NULL,
  `FelevNeve` char(12) NOT NULL,
  `TargyAzon` bigint(20) unsigned NOT NULL,
  PRIMARY KEY  (`KurzusAzon`),
  UNIQUE KEY `KurzusAzon` (`KurzusAzon`),
  KEY `TargyAzon` (`TargyAzon`),
  KEY `FelevNeve` (`FelevNeve`),
  CONSTRAINT `KURZUS_ibfk_1` FOREIGN KEY (`TargyAzon`)
REFERENCES `TARGY` (`TargyAzon`) ON DELETE CASCADE,
  CONSTRAINT `KURZUS_ibfk_2` FOREIGN KEY (`FelevNeve`)
REFERENCES `FELEVEK` (`FelevNeve`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin2;
```

```
--
-- Table structure for table `TANTERV`
--

DROP TABLE IF EXISTS `TANTERV`;
CREATE TABLE `TANTERV` (
  `TantervAzon` bigint(20) unsigned NOT NULL auto_incre-
ment,
  `Neve` char(48) default NULL,
  `ErvenyesTol` char(12) default NULL,
  `ErvenyesIg` char(12) default NULL,
  PRIMARY KEY  (`TantervAzon`),
  UNIQUE KEY `TantervAzon` (`TantervAzon`),
  KEY `ErvenyesTol` (`ErvenyesTol`),
  KEY `ErvenyesIg` (`ErvenyesIg`),
  CONSTRAINT `TANTERV_ibfk_1` FOREIGN KEY (`ErvenyesTol`)
REFERENCES `FELEVEK` (`FelevNeve`) ON DELETE CASCADE,
  CONSTRAINT `TANTERV_ibfk_2` FOREIGN KEY (`ErvenyesIg`)
REFERENCES `FELEVEK` (`FelevNeve`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin2;


--
-- Table structure for table `TARGY`
--

DROP TABLE IF EXISTS `TARGY`;
CREATE TABLE `TARGY` (
  `TargyAzon` bigint(20) unsigned NOT NULL auto_incre-
ment,
  `Roviditese` char(4) default NULL,
  `Neve` char(32) default NULL,
  PRIMARY KEY  (`TargyAzon`),
  UNIQUE KEY `TargyAzon` (`TargyAzon`),
  KEY `TargyRovidNeve` (`Roviditese`),
  KEY `TargyNeve` (`Neve`)
) ENGINE=InnoDB DEFAULT CHARSET=latin2;
```

```
--
-- Table structure for table `TARGY_BLOKKBAN`
--

DROP TABLE IF EXISTS `TARGY_BLOKKBAN`;
CREATE TABLE `TARGY_BLOKKBAN` (
  `TantervAzon` bigint(20) unsigned NOT NULL,
  `TargyAzon` bigint(20) unsigned NOT NULL,
  PRIMARY KEY  (`TantervAzon`,`TargyAzon`),
  KEY `TargyAzon` (`TargyAzon`),
  CONSTRAINT `TARGY_BLOKKBAN_ibfk_1` FOREIGN KEY (`Tan-
tervAzon`) REFERENCES `TANTERV` (`TantervAzon`) ON DELETE
CASCADE,
  CONSTRAINT `TARGY_BLOKKBAN_ibfk_2` FOREIGN KEY (`Targy-
Azon`) REFERENCES `TARGY` (`TargyAzon`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin2;


--
-- Table structure for table `VIZSGAALKALOM`
--

DROP TABLE IF EXISTS `VIZSGAALKALOM`;
CREATE TABLE `VIZSGAALKALOM` (
  `VizsgaAlkalomAzon` bigint(20) unsigned NOT NULL
auto_increment,
  `Fajta` char(2) default NULL,
  `Hely` char(8) default NULL,
  `Napja` date default NULL,
  `Kezdete` time default NULL,
  `Vege` time default NULL,
  `MaxLetszam` smallint(5) unsigned default NULL,
  PRIMARY KEY  (`VizsgaAlkalomAzon`),
  UNIQUE KEY `VizsgaAlkalomAzon` (`VizsgaAlkalomAzon`)
) ENGINE=InnoDB DEFAULT CHARSET=latin2;


--
-- Table structure for table `VIZSGAJA_VAN`
--
```

```
DROP TABLE IF EXISTS `VIZSGAJA_VAN`;
CREATE TABLE `VIZSGAJA_VAN` (
  `TantervAzon` bigint(20) unsigned NOT NULL,
  `TargyAzon` bigint(20) unsigned NOT NULL,
  `VizsgaAlkalomAzon` bigint(20) unsigned NOT NULL,
  PRIMARY KEY  (`TantervAzon`,`TargyAzon`,`VizsgaAlkalom-
Azon`),
  KEY `VizsgaAlkalomAzon` (`VizsgaAlkalomAzon`),
  CONSTRAINT `VIZSGAJA_VAN_ibfk_1` FOREIGN KEY (`Tanterv-
Azon`, `TargyAzon`) REFERENCES `TARGY_BLOKKBAN` (`Tan-
tervAzon`, `TargyAzon`) ON DELETE CASCADE,
  CONSTRAINT `VIZSGAJA_VAN_ibfk_2` FOREIGN KEY (`Vizsga-
AlkalomAzon`) REFERENCES `VIZSGAALKALOM` (`VizsgaAlkalom-
Azon`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin2;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS
*/;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLI-
ENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RE-
SULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNEC-
TION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;
```

Alkalmazási adatszabványok

Az adatdmodellezés során az adatok szerkezetének, kapcsolatainak, értéktartományainak (domain) gondos és pontos feltárásával és modellezésével számos korlátot állapítunk meg. Ezen korlátoknak alapvető jelentősége van a helyes adatbevitel elősegítésében, sőt kikényszerítésében.[207] A helyes adatbevitel szükséges (de nem elégséges) feltétele bármilyen adatbázis használhatóságának. A felismert és modellezett

207 L. az „A fogalmi valósághűség...” kezdetű bekezdéstől a Fogalmi modell c. részben, 27. old.

korlátok gépi érvényesítésének mellőzése a végeredmény minőségét nagymértékben lerontja.

Vannak azonban olyan korlátok is, amelyek gépi úton történő ellenőrzése, érvényesítése csak aránytalanul nehezen vagy egyáltalán nem oldható meg. Ilyenek például a nevek, címek írásmódja, mert olyan sokféle az érvényes lehetőségek száma, és olyan sokféleképpen lehet azt elrontani, hogy ennek gépi ellenőrzése elég reménytelen, illetve olyan mértékű részletezést követelne meg, amely indokolt az állami lakcímnyilvántartásban, de nem pl. egy tanulmányi rendszerben. Tipikusnak mondható hiba, amikor a titulust a név elejére írják, aminek eredményeképpen pl. az összes doktori fokozattal rendelkező kolléga ábécésorrendben a „d" betűhöz sorolódik, mint „dr. X. Y.". A változatosság kedvéért a „dr" hol kis, hol nagy „d", és vagy van utána pont, vagy nincs...

Ilyen esetekben felhasználói vagy alkalmazási adatszabványokat szükséges létrehozni, azaz formálisan elő kell írni a felhasználó számára (aki a géppel ellentétben gondolkodásra képes), hogy az adott adatféleséget milyen módon, milyen szabályok figyelembe vételével kell rögzíteni annak érdekében, hogy a későbbi visszakeresések eredményesek lehessenek. [F4260 pp. 142-144.] Az alábbiakban a személynevek és a címek írásmódjára vonatkozó, ún. írásszabványt mutatom be.

Személyek nevei (HALLGATO.Neve): elöl szerepel a családnév, nagy kezdőbetűvel, a többi betűje kisbetű. A családnév után egy darab szóköz, majd az első keresztnév szerepel, ugyancsak nagy kezdőbetűvel, a többi kisbetű. Esetleges második keresztnévre ugyanez vonatkozik. További keresztneveket nem rögzítünk. Esetlegesen előforduló külföldi nevek esetében az adott névre vonatkozó helyesírási szabályok vonatkoznak (pl. Price Derek deSolla prof.). A személy titulusát az utolsó keresztnév után, egy szóközzel elválasztva adjuk meg, és kizárólag az alábbiak egyike lehet: ifj., id., dr., özv., prof., PhD. Amennyiben egyéb titulus igénye merülne föl, az adatgazda intézkedik a szabvány bővítéséről. A nevek a magyar ábécé betűin kívül tartalmazhatnak pontot (a titulus végén) vagy kötőjelet (pl. Verseghy-Nagy). Egyéb karakterek esetén a név hivatalos írásmódja a mértékadó (pl. O'sváth). A magyar ékezetes magánhangzókat semmilyen körülmények között nem szabad mással helyettesíteni.

A lakcímek esete jóval összetettebb. Nem az írásszabvány része, de itt (is) hangsúlyozni kell a minimalitás elvét. Azaz a települések neveit *nem* adjuk meg minden cím részeként önállóan és egymástól függetlenül, redundáns módon, hanem modellezni kell a települések neveinek

és irányítószámainak a kapcsolatát. Ennek alapján az irányítószám alapján a település neve adódik (megfordítva is, alkalmazásfejlesztési, ergonómiai kérdés).

A településen belüli cím megadásakor első helyen áll a közterület neve a magyar helyesírás szabályainak és az adott közterület hivatalos nevének megfelelően, nagy kezdőbetűvel, a további betűk kisbetűk, rövidíteni nem szabad (pl. Kiss János altábornagy), hacsak maga a hivatalos név nem rövidített formájú (pl. Petőfi Sándor, ill. Petőfi). A név után egy szóköz következik, majd a közterület fajtájának a megnevezése, amely az itt felsoroltak valamelyike lehet az itt megadott írásmóddal: út, útja, u. (utca), sgt. (sugárút), s. (sétány), krt. (körút), tér, tere, körtér, dűlő, part, fasor, körönd, köz, lejtő, lépcső, major, puszta. A közterület fajtájának megadását egy szóköz követi, után következik a házszám (ha nincs házszám, a helyrajzi szám). A házszámot arab számokkal kell írni, közvetlenül utána pont van (pl. 3.). Összetett házszámokat kötőjellel, szóköz nélkül választunk el egymástól (pl. 4-6.) Albetétek esetén a házszámot a pont helyett „/” (per) követi, ezt pedig nagybetű, majd utána a pont (pl. 21/B.). Helyrajzi szám esetén a helyrajzi szám után egy szóközzel elválasztva a hrsz. rövidítést kell alkalmazni. Amennyiben többlakásos házról van szó, a következő elem a lépcsőház megadása (amennyiben van) következik, római számokkal, utána az lph. rövidítés (pl. III. lph.). Az emeletet is egy szóköz választja el, megadása ugyancsak római számokkal történik (pl. IV.), a végén pont szerepel, ha nincs ajtószám, vagy „/” (per) és az ajtószám arab számjegyekkel, a végén pont (pl. I/23.). Egyéb igény fölmerülése esetén az adatgazda intézkedik a szabvány megfelelő bővítéséről.

4. Alkalmazásfejlesztés

Kiválasztási eljárás

Sok esetben az alkalmazandó eszközök kiválasztása „magasabb szempontok" alapján, magyarán emberi gyengeségek szerint történik, amely kiválasztási eljárás nem tekinthető tudományos igényűnek, de többnyire még racionálisnak sem.[208] Normális esetben a megvalósítandó feladat sajátosságai, körülményei és adottságai az elsődlegesek, és ezek figyelembevételével, tervszerű megfontolás eredményeképpen választjuk a szükséges eszközöket. A kiválasztási eljárás során többféle, a választást segítő technikát alkalmazhatunk. Ezek lehetnek egyszerűbbek vagy bonyolultabbak, az ennek megfelelően kisebb vagy nagyobb ráfordításigénnyel, mint pl. a Franklin-mérleg, a hatásmátrix (POLANO-mátrix), a pontozótábla vagy a KIPA-módszer. [F4375 p. 7.6.7.3.][209]

A múlt század '60-70-es éveiben sokan foglalkoztak a több szempontú döntéstámogatás[210] kutatásával. Ezek segítségével a hatásmátrixból továbblépve számszerűsíteni lehet az egyes döntési szempontoknak való megfelelőséget, a szempontok egymáshoz viszonyított fontosságát. Hazai tudósok, Kindler József és Papp Ottó professzorok munkásságának eredménye az ún. KIPA-módszer (neveik kezdőbetűiből), amely igen komoly matematikai és módszertani eszközök felhasználásával még a döntéshozók kompetenciájának, valamint csoportos döntéshozatal esetén az egyetértés mértékének számszerűsítésére is kiterjed. [F4376]

Dacára a gondos elméleti alapozásnak és az igen nagyfokú megbízhatóságnak a számszerű MCDA módszereket ritkán alkalmazzák. Ezen eljárások alkalmazása ugyanis rendkívül idő-, munka- és információigényes, következésképpen költséges. Éppen ezért csak indokolt esetben használják ezeket a maguk elméleti-módszertani teljességében, amikor a döntési folyamat teljes egészének átláthatónak és (magának a módszernek legalábbis) támadhatatlannak kell lennie, pl. pályázatok elbírálásakor.

208 L. az Előtérben a technika, háttérben az elmélet, a modellalkotás és a szabványok c. részt, 67. old.

209 L. még http://www.tankonyvtar.hu/gazdasagtudomany/ » » controlling-gyakorlatban-080904-404

210 MCDA: Multi-Criteria Decision Aid

Esetünkben ilyen követelmény nem áll fenn, tehát bőségesen elegendő egy olyan egyszerűsített változat, a pontozó tábla használata, amely az indokolatlanul szubjektív elemeket képes kiszűrni a döntési folyamatból. Ez ráadásul az oktatásban használatos értékelési eljárásra hasonlít, ahol a diákok mindegyike vizsgázik minden tárgyból, és ennek megfelelően külön-külön értékelik őket. Itt a diákok felelnek meg a választási lehetőségeknek, az egyes tárgyak pedig a döntési szempontoknak, az adott diák adott tárgybeli vizsgája, annak értékelése pedig adott választási lehetőség adott szempontból történő értékelésének.

A pontozó tábla használatának van két előfeltétele.

Először is el kell végezni a választási lehetőségek előszűrését az ún. kizáró szempontok alapján. Kizáró szempontok azon szempontok, amelyeknek egy lehetőség vagy megfelel, vagy nem, utóbbi esetben kizárjuk a további értékelésből. Ilyen például a szoftver jogtiszta mivolta. Kizáró szempont lehet még egy soroló szempont is a hozzárendelt küszöbértékkel, pl. a licencdíj nem lehet több mint X forint.

Másodszor pedig el kell dönteni, hogy az ún. soroló szempontok alapján történő értékelést hány fokú skálán végezzük. A skála fokszáma nem lehet túl nagy, mert pl. egy százfokú skálán nem lehet elvárni a reális különbségtételt - mondjuk - a 63 és a 64 pontos értékelés között. A gyakorlatban jól beválik az oktatásban is megszokott ötfokozatú, esetleg és maximum a tízfokú skála alkalmazása. A háromfokú skála a hatásmátrix hármas (rossz, közepes, jó) színezéséhez való visszatérés lenne.

A pontozó táblával történő értékelés folyamata az, hogy minden értékelendő lehetőséget (amelyek a kizáró szempontok előszűrőjén átmentek) minden szempontból külön-külön értékelünk, pontozunk. A választási lehetőségek szerepelnek az egyes sorokban, a szempontok az egyes oszlopokban. A pontozás megtörténte után kiszámoljuk a sorösszegeket, és - jó esetben - a legmagasabb pontszámú lehetőséget választjuk, feltéve, hogy eléri az előzetesen kikötött küszöbértéket, mondjuk a maximális pontszám 75%-át. Ha nem éri el ezt a küszöbértéket, a kiválasztási eljárást sikertelennek ítéljük, és vagy további megoldási lehetőségeket kutatunk föl, vagy átgondoljuk szempontrendszerünket.

Van azonban két megvizsgálandó körülmény a végleges döntés meghozatala előtt. Az egyik: mi a teendő akkor, ha az összpontszámok az élmezőnyben túl kis mértékben térnek el egymástól, az értékelés becsülhető mérési hibáján belüliek a különbségek? A másik: az egyes értékelések során lehetnek kisebb-nagyobb bizonytalanságok, amelyek nyilván befolyásolják a megítélt pontok mennyiségét.

A bizonytalan pontértékeket a pontozás során meg kell jelölni. A végeredmény értékelésénél meg kell vizsgálni, hogy az élmezőny sorrendjét ezen pontértékek bizonytalansága befolyásolja-e, avagy sem. Ha nem, akkor az eredmény véglegesnek tekinthető, ha viszont befolyásolja, akkor nincs más lehetőség, mint a szóban forgó bizonytalanságok megszüntetése további adatgyűjtés és vizsgálatok (azaz további költség) árán.

Ha az élmezőny pontszámai közötti eltérés túl kicsi, kisebb mint az értékelési eljárás becsült mérési hibája, akkor két lehetőség van. Az egyik: újabb (szakmailag indokolható) szempontok figyelembe vétele mindaddig, amíg meggyőző különbség alakul ki az egyes lehetőségek között. A másik: tiszta lelkiismerettel mondhatjuk, hogy holtverseny alakult ki, és - most már - szubjektív rokonszenv alapján választunk.

A kiválasztási eljárás, pontosabban a döntési szempontok meghatározása ugyancsak modellezési eljárás. Mint ilyent, ezt is köti a modellezéssel kapcsolatos általános elvárásoknak való megfelelés, azaz a modellnek „jó”-nak kell lennie, az adott cél szempontjából „jól” kell leírnia a valóságot. Szempontjaink kiválasztása tehát meg kell feleljen a vizsgálat - esetünkben a döntéshozatal - céljainak, egyértelműnek, teljesnek (és természetesen ellentmondásmentesnek) kell lennie.[211]

Adatbázis-kezelő és környezetének kiválasztása

Ennek során kizáró szempontjaink lehetnek: jogtisztaság, fejlesztői/dokumentációs támogatás, legalább tábla szintű zárolás, értéktartomány kezelése, elsődleges és idegen kulcs kezelése, együttműködés http kiszolgálóval és külső programnyelvvel, valamint elvárt teljesítményküszöb, adott esetben a szabad szoftver mivolt (GNU-GPL licenc).

A soroló szempontok lehetnek: dokumentáció, referenciák (elterjedtség), hardverigény, futási/feldolgozási idők, adatbiztonság, támogatottság, megtanulhatóság, platformfüggetlenség, a teljesítmény skálázhatósága, CASE-eszközzel való együttműködés, a relációs elméletnek való megfelelőség mértéke, gazdanyelv fordíthatósága.

A legfontosabb, de nem az adatbáziskezelők tulajdonságain alapuló szempontok: a meglévő fejlesztői és üzemeltetési környezetbe való beilleszkedés, ár-teljesítmény arány.

211 L. 14. old.

Esetemben a jogtisztaság követelménye és a szabad szoftver mivolt követelménye mint kizáró szempontok érvényesülnek elsődlegesen. Utóbbi teljesülése az előbbi teljesülését is jelenti, ami a helyzetet egyszerűsíti. A szabad szoftver mivolt két okból fontos: egyrészt a biztonság, másrészt az ingyenesség okán.

A MySQL és a PostgreSQL közül ezek után már viszonylag könnyű a választás: a megtanulhatóság, a támogatottság (dokumentáltság), a referenciák (elterjedtség) okán már a MySQL felé billen a mérleg, a gazdanyelvi támogatás és a meglévő környezetbe való beilleszthetőség okán pedig egyértelművé válik a döntés. [F4302]

A biztonság két síkon is fontos tényező: egyrészt informatikai, másrészt pedig üzleti síkon. Az informatikai értelemben vett adatbiztonság a szoftver még ismeretlen, vagy már ismert, de még kijavítatlan hibáival van összefüggésben. Általános esetben a szabad szoftverek fejlesztői és tesztelői gárdája jóval nagyobb kapacitású, mint a zárt, üzleti szoftvereké, hiszen minden egyes felhasználója potenciálisan beletartozik.

Az üzleti értelemben vett biztonság pedig azt jelenti, hogy a jövőbeni felhasználás és támogatás vonatkozásában nem függünk egy olyan gazdasági társaságtól, amelynek üzleti érdekei nem szükségképpen esnek egybe a mi felhasználói érdekeinkkel. Magyarán: bármikor, üzleti vagy egyéb megfontolások alapján szüneteltetheti, akár végleg abbahagyhatja a szoftver fejlesztését, karbantartását, támogatását. Emellett nem lehetünk biztosak abban sem, hogy egy zárt, nem hozzáférhető forráskódú program nem tartalmaz-e olyan „szolgáltatásokat", amelyek számunkra legalábbis nemkívánatosak. Bizonyára nem véletlen, hogy egyes üzleti alapú licencszerződésekben még olyan kitételek is szerepelnek, miszerint a fejlesztő nem garantálja, hogy a szoftver a dokumentációnak megfelelően működik, hogy egyáltalán működik vagy működni fog a jövőben.[212] Szabad szoftver esetében a forráskód megléte és a forráskód bármilyen módosításának formális lehetősége ezen veszélyeket jelentősen csökkenti.

A MySQL választása alapvetően befolyásolja a gazdanyelv választását. Nyilván csak olyan vehető számításba, amely mind a MySQL adatbázissal, mind a http kiszolgálóval képes együttműködni. Esetünkben elterjedtsége, következésképp referenciája, dokumentáltsága alapján a PHP-re esett a választás.

212 pl. Xerox Ventura Publisher 2.0

Fejlesztőeszközök választása

A fentiek alapján a számomra egyáltalán hozzáférhető lehetőségek közül szabad szoftver mivolta alapján előnyben részesítettem a fabforce.net DBDesigner 4 for Linux eszközét a Sybase PowerDesigner eszközével szemben.

Kódolási szabványok

A szoftverfejlesztés minden körülmények között csapatmunka, még akkor is, ha a legszerényebb esetben a csapat egyetlen főből áll. Elsődleges fontosságú tehát az, hogy a fejlesztés eredményeképpen keletkező forráskód tiszta és világos legyen, amennyire csak lehetséges. Evvel nemcsak jelen és jövőbeni munkatársaink helyzetét könnyítjük meg, de a sajátunkét is. Igen ritkán fordul elő ugyanis az, hogy egy elkészült munkához a későbbiekben soha nem kell hozzányúlni.

Különösen igaz ez az információs rendszerek esetére, amelyek alapja egy adatmodell, amely a valóság egy kiválasztott részének működését tükrözi. A valóságos élet körülményei, feltételei pedig időről időre, kisebb-nagyobb mértékben megváltoznak, a felsőoktatásban is. Ennek következtében viszont szükségessé válik időről időre a megváltozott körülmények, az új igények miatt a meglévő kezelőprogramok átalakítása, kiegészítése, akár az eredeti adatmodell továbbfejlesztése is. Ez pedig csak akkor végezhető el hatékonyan, ha a meglévő programkódok, adatbázisdefiníciók, modellek, dokumentációk tiszta, világos logikával épülnek föl, és ezt formai sajátosságaik, külalakjuk is alátámasztja.

„A nehezen olvasható kód pedig nehezen karbantartható, és kínszenvedés benne a hibakeresés. A szegényes kódolási stílus a szakrételem hiányát mutatja.” [F4374 p. 3.] Igaz ugyan, hogy a tartalom az elsődleges és a forma a másodlagos, de az is igaz, hogy a forma lehet akár olyannyira célszerűtlen, amely visszahat a tartalomra is. Ezért szükséges külön meghatározni a kódolás során általánosan betartandó szabályokat.

A legfontosabb, ezért legelsőként említendő szempont az egységesség és következetesség követelménye. A következetes ragaszkodás a (közös) formai stílushoz fontosabb, mint maga a formai stílus. A következetesség és egységesség a kódban való könnyebb eligazodáson túl akár olyan előnnyel is járhat, hogy adott esetben a forráskód gépi úton is feldolgozható egyes, szükségessé vált átalakítások során.

A kódolás során fokozottan figyelemmel kísérendő területek a forráskód szerkezete, külalakja, az alkalmazott nevekre és azok formájára vonatkozó előírások, továbbá a dokumentáció kérdése. Természetesen a kódolási szabályok konkrét tartalma nem választható el az alkalmazott nyelvtől, esetleg befolyásolhatják azt még az alkalmazott segédeszközök is. Ezért bár a kódolási szabályok tényleges tartalma változhat, sőt változnia is kell a programkörnyezettől függően, a figyelembe venni szükséges szempontok és körülmények maguk alig változnak.

Esetünkben a kódolási szabályok értelemszerűen PHP-programokra vonatkoznak.

Szerkezeti kialakítás - tagolás

Az egyes fájlok kívánatos mérete két ellentmondó követelmény eredője. Egyrészt szerencsés, ha minél kevesebb darabszámú programfájlunk van, hiszen nehézkes mindig másik fájlt, fájlokat megnyitni és azokban keresgélni, azokat egymással összhangban javítani. Másrészt viszont a túl nagy méretek nehezítik az áttekinthetőséget és a funkcionális tagolást.

Az egyes programfájlokon belül az összefüggő kódblokkok méretére már sokkal könnyebb megfogalmazni azt a szabályt, hogy (ésszerű határokon belül) legyenek lehetőleg minél kisebbek a könnyebb áttekinthetőség érdekében. Gyakorlati tapasztalatok alapján azt a javaslatot tenném, hogy egy kódblokk semmiképpen se legyen hosszabb, mint kb. 60 sor, ennyi fér ki ugyanis a ma szokványos képernyőkre, és a kódblokk olyannyira zárt logikai egységet alkot, hogy azt mindenképpen célszerű egyszerre látni. (Még akkor is, ha az „igazi programozó 5 lap hosszú do ciklust tud írni anélkül, hogy belezavarodna”.[213]) Az egyes blokkok között egy üres sort ki kell hagyni, nagyobb jelentőségű blokkhatárokat (pl. függvény és eljárás kezdete, vége) egyéb módon is célszerű - egységesen - jelölni.

A kódban különféle megjegyzéseket lehet, sőt kell elhelyezni. Ezeknek szintén van elválasztó, blokkhatároló szerepük is. Amennyiben többféle lehetőségünk van a magyarázó megjegyzések (comment) jelölésére, ezen a területen is következetesnek kell lennünk. Alapvetően két különböző eset a rövid, pár szavas magyarázatok és a hosszabb, több soros magyarázatok esete. A legtöbb nyelvben ezek jelölésére külön eszköz áll rendelkezésre. A PHP esetében a két egymást követő perjel, ('//') vagy létra ('#') után a sor végéig lévő rész megjegyzésnek szá-

213 L. pl. http://szabilinux.hu/orlando_unix/igazi.html - az internetes irodalom és humor egyik gyöngyszeme.

mít. A két jelölési mód közül az egyiket kell kiválasztani, és ahhoz következetesen ragaszkodni.

A több sornyi megjegyzés jelölésére célszerű a megjegyzés kezdete ('/*') és vége ('*/') jelöléseket alkalmazni. Többsoros magyarázatokat általában az egyes fájlok elején, illetve az egyes eljárások és/vagy függvények elején szokás alkalmazni azok leírására.

Az eljárások és függvények kódblokkjának kezdetét érdemes feltűnőbben jelölni, pl. egy vagy két üres sor mellett még egy sor „sormintával" (pl. teljes sorhossznyi '#' karakter) is jelölni. Mivel ezek a deklarációs részben egymást követik, ezért jól tagolttá válik ez a rész. Mindenképpen érdemes viszont az egyes eljárások, illetve függvények végén, az utolsó utasítás-zárójel után megjegyzésben az adott függvény (eljárás) nevét feltüntetni annak függvény vagy eljárás mivoltával egyetemben (pl. '} # func Bejelntkezik'). Itt az esetleges paramétereket már nem soroljuk föl.

Szerkezeti kialakítás - a vezérlés logikája

A magas szintű programnyelvek lehetővé teszik, egyes esetekben kikényszerítik a strukturált forráskódírást. Ez rendkívül előnyös tulajdonság, igen nagymértékben könnyíti meg a forráskód későbbi olvashatóságát, illetve megértését. Érdemes alaposan kihasználni ezt a lehetőséget is.

A legelső, legfeltűnőbb és legkézenfekvőbb eszköz a behúzások lehetősége. Az egyes sorok beljebb való kezdése kifejezi a program vezérlési szerkezeteinek egymásba ágyazottságát, avval összhangban van. Ennek szokásos mértéke 2-4 szóköz, nekem személyes gyakorlatban a 3 szóköznyi lépcső vált be. A forráskódok szerkesztéséhez használt eszközök (pl. Midnight Commander) sajátosságai adott esetben szükségessé teszik annak előzetes eldöntését, hogy a behúzások csak szóközökkel valósíthatók-e meg, vagy tabulátorokkal is. Utóbbi esetben beállítás kérdése, hogy a tabulátor hány szóköznek felel meg, amely befolyásolhatja a forráskód megjelenítését más eszköz más beállításai esetén. Célszerű ezért a tabulátor karakterek használatának (nem a <Tab> billentyűnek) a mellőzése.

Az utasítás-zárójel olyan eleme a programnak, amely több utasítást is egyként fog össze, olyan esetekben, amikor valamilyen programszerkezeti elem adott helyén használják. Hiányában nem lehetne strukturált forráskódot írni. A C nyelvre hajazó programnyelvek esetében (mint pl. a PHP is) az utasítás-zárójel valódi zárójel, ezt a szerepet a kapcsos zárójel tölti be. Ennek használata során két kérdés merül föl: a) hasz-

náljuk-e akkor is, amikor ténylegesen csak egyetlen utasítást tennénk bele; b) külön sorba írjuk-e, avagy sem. Általában érdemes akkor is használni, amikor nem feltétlenül lenne muszáj, mert egységessége okán javítja a kód olvashatóságát, másrészt pedig ha a későbbiek folyamán mégis bővülne az utasítások száma a kérdéses helyen, biztosan nem felejtődik el az utasítás-zárójel kitétele, ha már eleve ott van. Ugyancsak érdemes a nyitó zárójelet az előtte álló vezérlési elem (pl. if, while) sorába írni, a záró párját pedig a vezérlési elem alá igazítani, evvel megtakarítunk egy sort, a kód kevésbé lesz töredezett.

A megfontoltan alkalmazott zárójelezés javítja az olvashatóságot. Különösen is érdemes zárójelezni az összetett logikai feltételeket, annál is inkább, mert a logikai operátoroknak ugyan megvan a precedencia-sorrendjük a matematikaiakhoz hasonlóan, de azok sokkal kevésbé köztudottak, még szakmai körökben is.

A programsorok hosszára általában van valamilyen korlát, amely az adott nyelv, értelmező, fordító saját jellemzője, és többnyire nem tudjuk, hogy pontosan mennyi is. Nem is szükséges, mert általában jóval több, mint amennyit célszerű lenne használni. A túl hosszú sorok ugyanis nehezítik az olvashatóságot. A 80 karakteres sorhossz talán túl konzervatívnak tűnik, de semmiképp sem érdemes jelentősen túllépni.

Mit tegyünk az olyan parancsokkal, amelyek túl hosszúak ahhoz, hogy elférjenek egy sorban? Logikus módon kell (érdemes) több sorba rendezni, azaz úgy, hogy az azonos vagy hasonló szerepű elemek kerüljenek egymás alá (pl. az összetett logikai feltétel egyes részei, a logikai operátorok a sor végén). Tipikusan hosszú, esetenként nagyon hosszú parancsok a többszörösen összetett logikai feltételt tartalmazó vezérlési elemek és az SQL-lekérdezések parancsai.

Az SQL parancsok szükség esetén több sorba való elosztását az SQL parancsok szerkezeti sajátosságainak a figyelembe vételével kell csinálni. Pl. a *select* utasítások esetén a *from,* a *where,* a *group by,* a *having* és az *order by* stb. záradékok álljanak a sor elején, ezek közül is a *where* esetében fölmerül az, hogy őt magát is több sorba kell rendezni, ez esetben a *where* utáni feltételeket egy lépcsővel beljebb kezdve.

Megkönnyíti az SQL-parancsok keresését, ha azokat mindig azonos nevű, szöveges típusú változókban helyezzük el az adatbáziskezelőhöz való továbbítás előtt, (pl. $sql, ha valamilyen okból kifolyólag egyszerre 2-3 parancsra is szükségünk van, akkor lehet használni pl. a $sql1, $sql2 változókat).

Változónevek

A változók nevei mindig legyenek értelmesek, utalván a programkódban játszott szerepükre. Az, hogy a változónevek angol vagy magyar nyelvűek legyenek, összetett kérdés. Ha nemzetközi fejlesztésről van szó, egyértelműen a nemzetköziség követelményeihez kell alkalmazkodni. Ha azonban hazai fejlesztésről van szó, sajnos akkor sem egyszerű a helyzet. Ugyanis anyanyelvünk ékezetes magánhangzóit a parancsértelmezők valamilyen megfontolásból nem fogadják el, tehát változónevekben általában csak az angol ábécé 26 betűjét, a számjegyeket és az '_' karaktert használhatjuk. Emiatt viszont avval a problémával kerülünk szembe, hogy az ékezetes magánhangzókat tartalmazó neveket vagy kerülnünk kell (kiváló anyanyelvi gyakorlat), vagy pedig egyszerűen ékezet nélkül írjuk azokat, ami ismét problémás lehet (gondoljunk csak a mára már anekdotává szelídült fókabél-főkábel esetre).

Mivel a szóköz nem lehet a változónevek része, az összetett szavakból vagy több szóból álló változónevekben a szóköz helyén alkalmazhatjuk az '_' karaktert ($sorok_szama), de szokás a vegyesen nagy- és kisbetűs írásmód is ($SorokSzama). Vannak akik az egyik, mások a másik írásmód mellett törnek lándzsát. Véleményem szerint ez is jól példázza az olyan helyzetet, amikor elég nehéz lenne a tárgyilagosságnak akár csak a látszatával eldönteni, hogy melyik megoldás a jobb. Válasszuk bármelyiket, de utána következetesen ragaszkodjunk hozzá minden munkatársunkkal egyetemben, legalábbis az adott munka során. Nem szabad szem elől téveszteni, hogy a változó- és függvénynevekben a kisbetű és a nagybetű különbözőnek számít, azaz *$neve* és *$Neve* két különböző változó.

Az értelmes nevek használata alól van egy kivétel: a jelentéktelen szerepű, „egyszerhasználatos” változók, segédváltozók, különösképpen egyszerű ciklusok ciklusváltozói esetében használjuk a $i, $j ($k, $l, $m, $n) változóneveket, amelyek még a Fortran korából öröklódtek, aholis a változónév első karaktere implicit típusdefiníció volt. Hasonlóképpen szöveges típusú segédváltozóknak nyugodtan használhatjuk a $s, $s1 stb. mintájú változóneveket.

Ha adott jellegű változóval kapcsolatban többféle szerepkörre van szükségünk, elnevezési rendszerünk következetes legyen minden ilyen esetben (pl. $darab, $darab_min, $darab_max, $darab_sum).

Helyes szokás az állandók neveit csupa nagybetűvel írni, evvel is megkülönböztetve őket a „közönséges” változókból.

HTML-kód előállítása

HTML-dokumentumokat alapvetően háromféle módon lehet előállítani a PHP segítségével. Erre az esetre is vonatkozik az, hogy az egységesség, következetesség fontosabb annál, hogy melyik változat a jobb (vagy jobbnak tartott).

Az első lehetőség, amikor az ügyfélnek elküldendő HTML minden egyes sorát a PHP program állítja elő, azokat soronként egy print vagy echo utasítással kiíratva. Igen nehézkes megoldás, a sok print/echo nehezíti a html-kód olvasását, módosítását.

A második lehetőség a PHP és a HTML kölcsönös egymásba ágyazhatóságát használja ki. Ebben az esetben a HTML dokumentum változó elemei helyén szerepelnek rövid PHP függvényhívások, amelyek az adott helyen lévő szövegelemet vagy szövegelemeket előállítják.

Mindkét megoldásnak hátránya, hogy körülményessé teszi a PHP-programozók és a HTML-szerkesztők együttműködését, mert egymás kódját kellene tiszteletben tartaniuk úgy, hogy esetenként kénytelenek hozzányúlni a másik szakember kódjaihoz.

A harmadik lehetőség sablonok használatára épül, ez esetben válik szét a legteljesebb módon a PHP programkód és a HTML dokumentumkód egymástól. Ha nem az első két lehetőség valamelyikét alkalmazó, már elkezdett munkát öröklünk, mindenképpen ez utóbbit érdemes választani.

Dokumentáció készítése

A megfelelő tartalmú és formájú dokumentáció nélkülözhetetlen, elemi része a munkának. Hiányában minden későbbi munkafázis nehezebb, körülményesebb, akár (gazdasági, pénzügyi értelemben legalábbis) lehetetlenné válik. A dokumentáció lehet felhasználói és fejlesztői. Az előbbi gondos elkészítése fontos feladat, de nem az alkalmazásfejlesztés szerves része, ellentétben az utóbbival.

A fejlesztői dokumentációnak is két fő területe van. Az egyik a külső, API-dokumentáció, a másik a belső dokumentáció. Az API[214] az alkalmazásprogramozási felület: egy program vagy rendszerprogram eljárásainak és függvényeinek (szolgáltatásainak) és azok használatának dokumentációja, amelyet más programok felhasználhatnak. Az API-dokumentáció segítségével lehetséges egy programrendszer szolgáltatásait használni anélkül, hogy annak belső működését ismernie kellene a programozóknak. Formálisan megadja, hogy adott eljárás vagy függ-

214 application programming interface

vény milyen szolgáltatást nyújt, milyen bemenő adatokat vár és milyen eredményt vagy eredményeket ad vissza.

A belső dokumentáció a programkódban elhelyezett magyarázó megjegyzésekkel annak működési logikájának megértését hivatott elősegíteni, az alkalmazott algoritmus és az alkalmazott kódolástechnikai megoldás rövid magyarázatával és szükség szerinti indoklásával. Nem tévesztendő szem elől, hogy ez szakember kollégáknak (esetenként akár saját magunknak) készül. A semmitmondó megjegyzések csak az áttekinthetőséget zavarják, fölöslegesek (pl. az '$i++;' utasítást fölösleges megmagyarázni '# $i növelése' formájában. Ha ez nem nyilvánvaló a kolléga számára, akkor nem kolléga. Arra azonban nem utal, hogy miért is szükséges ez a növelés. [F4374 p. 25.]

Ha a programfejlesztés és a dokumentálás térben és időben szétválik, jó eséllyel sosem készül el, vagy ha mégis, problémás lesz a dokumentáció és a kód egymásnak való megfelelősége. (Vö. „az igazi programozó programot ír, és nem dokumentációt”.[215]) Ezért a dokumentálás lényegi részét, ha nem teljes egészét a programkód kell tartalmazza a megfelelő magyarázatok formájában.

A megfelelő, célszerű és egységes formában készült megjegyzések alkalmasak arra is, hogy azokat részben vagy egészben gépi úton feldolgozva készüljön a fejlesztői, illetve a felhasználói dokumentáció. Ennek van már sokoldalú gépi támogatása is, a PHP esetében ilyen eszköz például a PhpDocumentor[216]. Képes egyrészt felismerni az alapvető fontosságú programnyelvi elemeket (osztályok, függvények, tulajdonságok, öröklődés stb.), valamint az előírt formájú (DocBlock) megjegyzésblokkokat. Ezek alapján létrehozza a DocBook XML formátumú dokumentációt. Előnye, hogy a programfejlesztés és a program dokumentálása funkciókat a lehető legközelebb hozza egymáshoz, jelentős mértékben megkönnyítve a fejlesztők munkáját, és elősegítve a dokumentáció naprakész mivoltát. Ilyen körülmények között nem szükséges ugyanis a dokumentáció külön javítgatásával foglalkozni, hanem az újra generálható az értelemszerűen megváltozott programkódbeli magyarázatok alapján.

215 Az internetes irodalom egyik gyöngyszeme Az igazi programozó. Magyarul megtalálható pl. http://szabilinux.hu/orlando_unix/igazi.html

216 http://www.phpdoc.org

SQL-kód

Külön kell foglalkozni az SQL-parancsok írásmódjával. Ez fölmerült már a PHP-kódolás formális előírásai között is, de a PHP szemszögéből nézve. Külön kell foglalkozni az adatbázist ténylegesen kezelő adatmanipulációs utasítások (*select, update* stb.) célszerű formájával, valamint az adatbázisban használandó névkonvenciókkal.[217] Erre is vonatkozik az egységesség elsődleges követelménye. Formai sajátosságként követhető az a szabály, hogy a táblanevek csupa nagybetűvel, az oszlopnevek nagy kezdőbetűvel írandók, az egyebek (pl. index, megszorítás) nevei pedig csupa kisbetűvel. Fölösleges az oszlopok neveit (mezőnevek) a tábla nevével vagy annak rövidítésével kezdeni, hiszem minden SQL-utasításban kiírható, esetenként kiírandó a „teljes nevük", „TÁBLA.Oszlopnév" formában. Modellezési szintű szabály a (technikai) szinonimák, homonimák elkerülése, időben való kiszűrése.

Adatellenőrzés

Ugyan nem formai, hanem tartalmi lényeg, de szót kell ejteni egy igen fontos szabályról: a felhasználótól érkező adatot mindig ellenőrizni kell minden lehetséges (és lehetetlennek tűnő) szempontból is, annak fogadása után a legelső lépésében. Ennek elmulasztása nemcsak az adatbázisban fellelhető selejtes adatok arányát növeli, nemcsak programjaink működőképességét veszélyezteti, de kiemelt biztonsági kockázatot is jelent (vö. SQL-befecskendezéses, puffertúlcsordulásos támadások). Ezen ellenőrzésnek vannak meghatározható formai elemei is: előírjuk, hogy minden felhasználótól származó adatot egy ellenőrző függvényen keresztül szabad csak fölhasználni.

Változatok követése

Változatkezelő vagy változatkövető rendszer alkalmazása nélkülözhetetlen csoportmunkában történő fejlesztés esetén, de igen jó szolgálatokat tehet egyéni munka során is. Működésének lényege, hogy az egyes változásokat nyomon követhetővé és visszafordíthatóvá teszi. Csapatmunkában több fejlesztő is párhuzamosan dolgozhat az adott projekten, úgy, hogy a párhuzamos munkavégzés biztonságosan zajlik, anélkül, hogy egymás munkáját felülírnák a munkatársak.

217 L. a „„...nincs egy TÉNYLEG jó..." kezdetű bekezdést a Tervezési segédeszközök választása c. részben, 115. old.

A változatkövetés történetét igen röviden foglalja össze Schlossnagle: „A változatkezelő vagy változatkövető rendszerek 'szabványa' a nyílt forráskódú programok között ma a CVS (Concurrent Versioning System). A CVS az RCS (Revision Control System) bővítéseként jött létre. Az RCS-t Walter Tichy írta a Purdue University-n 1985-ben; ez is egy korábbi rendszer, az ATT Labs cégnél 1975-ben kidolgozott SCSS (Source Code Control System) javítása volt. Az RCS-t szerzője azért írta, hogy lehetővé tegye, hogy többen dolgozzanak ugyanazon a fájlhalmazon, egy bonyolult zárolási rendszer segítségével. A CVS az RCS-re épül; megengedi, hogy egy fájlnak több tulajdonosa legyen, lehetővé teszi a tartalmak automatikus összeolvasztását, a forrásfa felépítését, illetve azt, hogy egyszerre több felhasználó rendelkezzen írható példánnyal a forráskódból.” [F4374 pp. 194-195.]

Fölmerülhet a kérdés, hogy a rendszeres időközönként (naponta) végzett biztonsági mentések nem elegendők-e, nem felelnek-e meg ennek a célnak. A válasz az, hogy bár a biztonsági mentésekre feltétlenül szükség van (mégpedig lehetőleg lépcsőzetes mentésre), de a mentés önmagában nem helyettesíti a változatkövetést, hiszen a mentéshez pl. nem tartozik semmilyen dokumentáció, amely az időközben elvégzett változtatásokkal bármilyen kapcsolatban lenne. Ugyanakkor az egyes, mégoly aprónak tűnő változtatások is nehezen kiszámítható hatással lehetnek a végtermék működőképességére, ezen hatás áttételessége, kiszámíthatatlansága pedig a forráskód méretének exponenciális jellegű függvényével jellemezhető.[218]

„Sajnos nem lehet megmondani, hogy egy adott alkalmazás mikor 'működik' és mikor 'hibás' - a működőképességet ennél sokkal finomabb skálán mérik, valahol a fut és a leállt közötti szürke sávban helyezhető el. Még egy apró beállításmódosítás is olyan rafinált hatásokat válthat ki, amelyek napokig (vagy hetekig) nem láthatók. (...) minden valamirevaló változatkövető rendszer számon tartja bármely fájl módosításait, valamint azt, hogy ki, mikor és miért módosította azt. Ezek az adatok leírhatatlan értéket képviselhetnek egy összetett rendszer esetében, főleg, ha több ember végzi a karbantartását. (...) a fájlok tetszőleges korábbi állapotához 'vissza lehet tekerni', és az akkori és a jelenleg érvényes beállítások közötti különbségek elemezhetők.” [F4377 p. 69.]

Valamely változatkövető rendszer használatára mindenképpen szükség van. Előnyös lehetne, ha az alkalmazott, vagy alkalmazandó CASE-

218 L. az „A tervezés fontossága...” kezdetű bekezdéstől, 95. old.

eszköz ezt a feladatot is magára vállalná, tekintettel az adatbázistervezés különleges igényeire, de ez nem feltétlenül teljesíthető elvárás.

5. A hatékonyság vizsgálata

„A hatékonyság relatív és nem abszolút fogalom. (...) A hatékonyság klasszikus mércéi többek között a következők: a folyamat tényleges energiafogyasztása és az elméleti minimum hányadosa...” [F4340 p. 112.] Nyilvánvaló, hogy egy számítógépes program(rendszer) esetében a hatékonyság („hatásfok”) gazdasági életből vagy fizikából vett fogalma nem alkalmazható, ha mégis megtesszük, akkor jelentése és jelentősége semmitmondó lesz. Szükséges tehát, hogy a fogalmat a körülményeknek megfelelően értelmezzük.

A fizikai fogalom azt fejezi ki, hogy egy eszköz a befektetett energia hány százalékát fordítja ténylegesen az elérni kívánt célra (a többi az adott szempontból veszteség). Ezt a fogalmat úgy lehet általánosítani, hogy az összes ráfordított erőforrás és a kívánt cél elérésére ténylegesen hasznosuló erőforrás arányáról beszélünk. Szoftverek esetében még ez a megközelítés is nehezen értelmezhető.

Ha azonban úgy közelítünk a hatékonyság fogalmához, hogy az elméleti tökéletesség (100,00%) hányadrészét éri el az adott eszköz, rendszer, akkor, ha nincs is könnyű dolgunk a szoftverek esetében, de legalább már nem reménytelen a helyzetünk. Ilymódon a hatékonyság, hatásfok kérdését minőségi kérdéssé transzformáltuk, ez pedig a szoftverek esetében is értelmezhető, ha nem is olyan könnyen és egyszerűen, mint az energetikai hatásfok.

Minőségi elvárások

A szoftverek minőségének értékelése során felhasználható, sőt felhasználandó szabványos szempontrendszert a Szoftvertermékek értékelése, minőségi jellemzők és használatuk irányelvei című szabvány[219] írja le. Ennek alapján a szakirodalom a minőségi jellemzőket hat csoportba sorolja be. Ezek a következők: funkcionalitás, megbízhatóság, használhatóság, hatékonyság, karbantarthatóság, hordozhatóság. [F4378 pp. 377-380.]

1. Funkcionalitás: Képes-e a meghatározott és elvárt szolgáltatást vagy szolgáltatásokat nyújtani? A valóságos folyamatoknak való meg-

219 MSZ ISO/IEC 9126:2000

felelés, a specifikáció szerinti működés, adat- és programhelyesség, szabványoknak, jogi szabályozásnak megfelelő működés, szállítótól független továbbfejlesztési lehetőség.

2. Megbízhatóság: A működőképesség megtartása meghatározott körülmények között meghatározott ideig. A hibatűrés, helyreállíthatóság, hibaelőfordulás, rendelkezésre állás mértéke és jellemzői.

3. Használhatóság: A felhasználóknak milyen általános és specifikus ismeretekre van szükségük a használathoz? Milyen az érthetőség, tanulhatóság és üzemeltethetőség?

4. Hatékonyság: Milyen a szoftver alkalmazásával biztosított teljesítmény és a használathoz szükséges erőforrások viszonya? Milyen az adatfeldolgozás sebessége, mekkorák a tipikus, illetve a leghosszabb válaszidők? Milyen a hardvererőforrás(ok) kihasználása, az alkalmazott hardver- és szoftvermegoldások korszerűsége és célszerűsége?

5. Karbantarthatóság: Mekkora a szükséges módosítások erőforrásigénye? Milyen a tesztelhetőség, az adaptálhatóság, a telepíthetőség, a helyettesíthetőség, a karbantartás, a mentés és a helyreállítás (visszatöltés) erőforrásigénye, a rendszerdokumentáció minősége?

6. Hordozhatóság: A rendszer egy adott hardver- szoftver-, illetve szervezeti környezetből egy másik környezetbe történő áttelepítésének nehézsége és kockázata mekkora?

Minőségi elvárásokkal kapcsolatos szabványok

Magyarországon minőségbiztosítási rendszernek hívták az MSZ EN ISO 9001:1996 szabvány alapján kialakított rendszereket. Az ISO 2000-ben jelentette meg az ISO 9001 harmadik változatát, amelyet a Magyar Szabványügyi Testület 2001-ben vett át. Az új szabvány alapján megvalósított rendszereket nevezzük minőségirányítási rendszereknek. Az ISO 9001 szabvány negyedik változatának kiadására 2008-ben került sor, amit a Magyar Szabványügyi Testület 2009 elején adott ki magyar fordításban.[220] Ez az új szabvány alapvetően nagy változásokat nem tartalmazott, hanem inkább pontosításokkal és magyarázatokkal segítette az előző verzió könnyebb érthetőségét, jobb felhasználását.

Az IEEE 1012-1986 (Szoftvervizsgálat és -értékelés), valamint az IEEE 1028-1988 (Szoftver átvizsgálás és auditálás) szabványok alapján az SQA[221] fogalmazott meg szabványos követelményeket.

220 MSZ EN ISO 9001:2009
221 Software Quality Assurance

Mérések tervezésének szempontjai

Egy kísérletben szándékosan változtatjuk a kiszemelt változókat annak érdekében, hogy megfigyeljük ezen változások hatását a minőséget meghatározó változókra. A kísérletek matematikai, statisztikai eszközök felhasználásán alapuló megtervezése hatékony módja a kísérletek eredményeinek elemzésére oly módon, hogy a kapott adatok tényszerű következtetések levonását tegyék lehetővé, illetve támasszák alá. A kísérletterv tartalmazza részletesen a különféle beállításokat, sorrendet, és - terv lévén - még a tervezett kísérletek megkezdése előtt rendelkezésre kell állnia.

A jól meghatározott kísérlettervek maximalizálják a kísérletek során kinyerhető információt, illetve minimalizálják a kívánt eredmény eléréshez szükséges kísérletek (mérések) számát. A vizsgálni kívánt paraméterek összes lehetséges kombinációját nyilván igen gazdaságtalan (gyakorlatilag lehetetlen) kipróbálni, ezért a lehető legkevesebb darabszámú beállítás mellett vizsgáljuk az elért eredményeket.

A lehetséges felhasználási módok számosak, a különféle lehetőségek közötti választástól a minőséget meghatározó kulcstényezők kiválasztásán keresztül a lokális szélsőértékkeresésig. 1990-ben az utóbbi területen készítettem szakdolgozatomat az ELTE TTK-n.

A statisztikai megfontolások alapján tervezett kísérletek időt és erőforrást takarítanak meg, és csökkentik a kockázatot. R. Fisher volt az első, aki kismintás kísérletek eredményeinek kiértékelésére szolgáló módszereket alakított ki. A mai viszonyok között, a számítógépes szimulációk esetében új kísérlettervek és -módszerek is számításba jönnek. [F4346 pp. 61-65.] A statisztikai elemzések lehetőségeit tankönyvi részletességgel és alapossággal tárgyalja magyar nyelven pl. a Falus - Ollé szerzőpáros [F4341].

A legfontosabb követelmények: a világosan és pontosan meghatározott cél (a kiválasztott minőségi jellemzők célértéke, kis ingadozása); az elegendő informativitás (a minőség összes jellemzőjének számbavétele); az egyértelmű eredmények (biztosan azt mérjük, és azt a paraméterfüggést határozzuk meg, amit szándékoztunk); a minimális ráfordítás; a meghatározott érvényességi terület.

Egy kísérletterv is adhat használható eredményt, de sokkal valószínűbb, hogy többre is szükség lesz ahhoz, hogy a teljes választ megadjuk. Az elvégzett kísérletek elemzéséből ugyanis folyamatosan lehet következtetni a vizsgált folyamat működésére, ez pedig befolyásolhatja a további kísérleteket (a kísérlettervezés aktív módszere).

Esetünkben számos olyan körülmény merül föl, amely a mérést egyszerűbbé teszi. Nem ipari környezetben, működő rendszeren kell elvégezni a méréseket, továbbá a kísérlet időigénye nem túl jelentős, tehát a kísérletszám tág határok között gyakorlatilag költségfüggetlen módon választható meg. Mivel az elvégzett kísérletek értékelése során ismereteink bővülnek, jobban megismerjük a mért jelenség természetét, a további kísérleteket az újabb ismeretek alapján tervezhetjük. Ez jóval hatékonyabb eljárást eredményez.

Törekednünk kell arra, hogy maga a mérés a lehető legegyszerűbb legyen. Az egyszerűség követelménye alapján tehát igyekeznünk kell, hogy „jó kérdést tegyünk föl", azaz a mérendő értéket úgy válasszuk meg, hogy az számunkra előnyös legyen: minél közelebbi kapcsolat legyen a mért adatok és az azokat befolyásoló tényezők között, továbbá minél kevesebb és minél kisebb jelentőségű és mértékű zavaró hatás léphessen föl.

Nem az egyes paraméterek számszerű hatásának kimérése, a válaszidőt leíró függvény és paramétereinek matematikai meghatározása a célom, hanem a rendszer működésének az optimális (vagy azt megközelítő) tartományba[222] juttatása, illetve az alapfeltevés megerősítése ill. cáfolata. A válaszidőket meghatározó fontos tényezők úgyis ismertek, és nagymértékben a technikai feltételek hatása alatt állnak. Az alapfeltevés az, hogy a jobb adatmodellezés lényegesen jobb eredménnyel jár.

Mérési terv

Azt, hogy egy adatbázis mennyire terhelhető (stressz teszt), azaz hogyan képes megbirkózni szélsőségesen nagy igénybevétellel, számos tényező befolyásolja. Ezen tényezők közül a hardver fizikai teljesítményét lehetne elsőként említeni. A hardver teljesítményének növelése a nagyobb adatbázis-teljesítmény elérése érdekében nem más, mint a közismert nyers erő („brute force") alkalmazásának egy példája. Másképpen mondva: akinek több pénze van, annak nagyobb teljesítmény fog a rendelkezésére állni.

Vannak azonban ennél finomabb és jóval olcsóbb módszerek is, amelyekkel az adatbázis teljesítménye jelentősen növelhető.

Ilyen például az operációs rendszer, az adatbáziskezelő és az alkalmazás maga, mint a szoftverkörnyezet leginkább meghatározó elemei.

222 L. az „Egyik szóhasználati mód..." kezdetű bekezdést a Rendszer c. részben, 13. old.

Az első kettő egy-egy jól meghatározott halmazból választható legfontosabb műszaki és kompatibilitási paramétereik alapján. Néhány technikai körülmény hatását a teljesítményre korábban már vizsgáltam. [F4369]

A harmadik tényező, az alkalmazás esetében számos lehetőség van a teljesítmény növelésére. A programozási nyelv és eszközök kiválasztásán túl két lényegi terület határozza meg az elérhető teljesítményt. Ezek az algoritmus és a programkódolás minősége, illetve hatékonysága. Adatbázisok esetében az algoritmus jóságába az adatmodell jósága is beleértendő, mint a talán legfontosabb, szükséges (de önmagában nem elégséges) körülmény.

Számos olyan tényező van, amely nemcsak befolyásolhatja, de befolyásolja is a mérést. Ezek közül néhány, esetünkben szóba jöhető tényezőt - a teljesség igénye nélkül - a következőkben ismertetek.

Mivel a mérést egy működő számítógépes környezetben végezzük, az operációs rendszer minden egyéb tevékenységét figyelembe kellene venni. Például egy fájl mentése a helyi merevlemezre mindenképpen igényel valamennyi (bár esetenként nagyon kicsi, de mindenképpen nullánál több) időt. Ez véletlen hibának tekinthető, mert a merevlemez író-olvasó fejének mozgása és a lemezpuffer pillanatnyi állapota határozza meg.

Ugyancsak befolyásolja a mérési eredményeket a méréstől függetlenül fölmerülő egyéb hálózati adatforgalom. Ez csökkenthető lehet, ha az alhálózatot a mérések idejére lezárjuk minden egyéb adatforgalom elől, de meg nem szüntethető, mert véletlenszerű elemek is befolyásolják, ilyen pl. az ethernet-adatkeretek esetleges ütközésének problémája.

Végül, de nem utolsósorban maga a mérés is befolyásolja önmagát, mivel az adott számítógépes tevékenységet számítógépes tevékenységgel mérjük, ami valamilyen egyszerűbb vagy összetettebb program futtatását követeli meg, tehát a rendelkezésre álló összes erőforrás egy részét fölemészti.

Mindezeken kívül vannak további hibák is, például az észlelési és a számítási hibák is, amelyek hatását valamilyen módon számításba kell venni.

Mit és hogyan érdemes mérni

Mivel arra vagyunk kíváncsiak, hogy adatbázisunk mennyire stressztűrő, azaz mennyiben képes megbirkózni a kiugróan nagy igénybevétellel, mindenképpen válaszidőt, válaszidőket kell mérnem olyan tevé-

kenység során, amely önmagában kiugró terhelést jelent. Esetünkben két ilyen terület van: a kurzusokra és a vizsgákra történő jelentkezés.[223]

Ezek közül a vizsgákra való jelentkezés vizsgálatát választottam, szubjektíven. Ezt azért tehetem meg, mert lényegét tekintve nincs érdemi (nagyságrendi) különbség a két tevékenység között. Ennél jóval érdekesebb kérdés az, hogy mit is kellene mérni. A pontos válaszidőt minden egyes hallgató minden vizsgajelentkezésének esetében? A próbálkozások számát és a válaszidőket? Adott időtartamra jutó sikeres és sikertelen jelentkezési kísérletek számát?

Eredetileg ABC-elemzést terveztem csinálni, amikoris a válaszidők alapján három halmazba sorolhatók az eredmények, amelyek minősítése „nagyon jó", „elfogadható" és „elfogadhatatlan". De melyek azon értékek, amelyek az egyes halmazok határainál húzódnak? Hány másodperces válaszidő számítson „nagyon jó"-nak, és hány másodperc után tekintsük „elfogadhatatlan"-nak? Mivel ezek meglehetősen szubjektív minősítések, ezért a tényleges időérték-tartományok hozzárendelése nehezen lenne megindokolható, ráadásul a határvonalak amúgy sem lehetnének merev határvonalak (mint a 0 a negatív és a pozitív számok között pl.), másrészt pedig a körülmények függvényében idők folyamán igencsak változhatnak. Célszerűnek látszik tehát a fentinél egyértelműbb minősítési eljárás választása. Az első mérések adatai alapján be is bizonyosodott, hogy a helyzet sokkal egyszerűbb.

A végfelhasználók szempontjából nincs jelentősége a válaszidő 2-4 tizedesjegy pontosságú tényleges értékének. Ami a végfelhasználók szempontjából lényeges, az az, hogy a) a válaszidők elfogadhatók-e egyáltalán; b) a jelentkezési kísérletek mekkora hányada sikeres. Természetes, hogy az a) kérdésre adandó válasz továbbra is erősen szubjektív, de sokkal kevesebb problémát vet föl, mint az ABC-elemzés halmazaiba való besorolás. Így végül az átlagos és a maximális válaszidő, továbbá a sikertelen kísérletek számának meghatározását választottam jellemző adatként, és ezeket mértem azon megfontolás alapján, hogy amennyiben a válaszidők értékei és a sikertelen kísérletek aránya „elegendően" alacsonyak egy, a ténylegesen használt rendszerénél szerényebb erőforrás-környezetben, akkor az adatmodellezés jóságának és fontosságának szerepét sikerül ha nem is bizonyítani, de jelentősen valószínűsíteni.

223 L. 126. old.

Hogyan történjen a mérés

Mivel csúcsterhelési helyzetet szeretnék vizsgálni, mindenképpen nagyszámú hallgató vizsgajelentkezését kell lebonyolítani tesztkörülmények között, azaz a hallgatók személyes közreműködésének mellőzésével. Ennek megoldására több lehetőség is kínálkozott.

Az egyik lehetőség az lenne, hogy a hallgatókat parancsfájlok (shell script) „személyesítik meg". Ez esetben a mérés menete a következő: a parancsfájl egy parancssori böngésző segítségével bejelentkezik a teszt-rendszerbe, lekéri a saját vizsgaidőpontjait, mégpedig minden vizsgaköteles tárgyának minden vizsgaidőpontját. Ezek közül valamilyen algoritmus alapján kiválaszt tárgyanként egyet-egyet, majd azt mint jelentkezést elküldi. Az erre kapott választ megvizsgálja, hogy minden tárgyból sikeres volt-e a jelentkezés. Amelyekből nem volt sikeres a jelentkezés - mert közben a kiválasztott vizsgaidőpontra elfogytak a szabad helyek -, azon tárgyakból újabb időpontot választ, és ezt mindaddig ismétli, amíg minden tárgyból nem sikerült egy helyet találnia.

Ha elegendően nagyszámú alkalom és elegendő férőhely van a lehetőségek között, akkor véges sok kísérlet során minden tárgy valamely vizsgaalkalmára sikerülnie kell a jelentkezésnek. Célszerű a lehetőségek közül való választás során a véletlenszerűséget beépíteni, különben nem kizárható a végtelen ciklusba való keveredés lehetősége. Ezen parancsfájlokat a szükséges számú számítógépen elhelyezve és futtatva lehet szimulálni a hallgatók vizsgajelentkezését. A parancsfájlokat futtató gépeken pedig rögzíthetjük az egyes jelentkezések kezdeti és befejezési időpontjait a mérés kiértékeléséhez.

A parancsfájlokat futtató gépek számát alapvetően az határozza meg, hogy ezek közül egy adott gép egyidejűleg hány parancsfájl futtatására képes. Az adatbázis terhelésének mértéke szempontjából közömbös, hogy a kérések hány másik gépről érkeznek, a lényeg az, hogy hány kérést kap adott idő alatt.

Mivel ez elég bonyolult és összetett parancsfájlokat követelne meg, ráadásul több gép esetén a végrehajtás időzítése is problémás, olyan más megoldást kerestem, amely a mérési feladat elvégzésének szempontjából egyszerűbb, ugyanakkor azonban a mért eredményt kismértékben ronthatja, de semmiképpen sem javíthatja.

A fent vázolt folyamatot egyszerűsítve - az adatbázis- és alkalmazáskiszolgáló terhelését valamelyest növelve - azt a lehetőséget választottam, hogy a hallgatókat jelképező parancsfájlok semmilyen értékelés nem végeznek, a lehetséges vizsgaalkalmak közüli véletlen választást

(mindaddig ismételve, amíg nem sikerül minden tárgyból valamely alkalomra jelentkezni) is a kiszolgáló oldal végzi, beleépítve a vizsgajelentkezéseket lebonyolító teszt-alkalmazásba. Így tehát a parancsfájl megfelelően paraméterezett parancssori böngészővel meghívja a teszt-alkalmazás index.php programját, és ha véges időn belül választ kap, akkor az a sikeres vizsgajelentkezések tényleges időpontjait tartalmazza, amelyet ment egy helyi fájlba. A parancssori böngésző naplózza a kapcsolódás időpontját, illetve a kapott válasz helyi fájlba mentésének (rossz esetben az időtúllépés bekövetkeztének) időpontját. A kettő különbsége adja a mérendő értéket.

A használt mérési eljárás során az adatbázis szerkezete közvetlenül az optimálisnak tartott fogalmi modellen alapul, azaz a logikai szintű tervezés során nem alkalmaztam semmilyen, a hatékonyságot érdemben és közvetlenül növelő szerkezeti átalakítást. Ilyenre azonban szükség esetén van lehetőség, mint azt a logikai tervezés c. részben[224] bemutattam.

A mért változat eredményein alapuló becslésem szerint ez a megoldás nem okozna számottevő időeltolódást, az adatbázist érő terhelési csúcsot ugyanakkor jelentősen simítaná. A megelőző változatbeli körülmények között elvégezve a mérést azonban azt találtam, hogy erre a megoldásra - egyelőre legalábbis - nincs szükség.

Mérési környezet

A mérés személyi számítógépes környezetben történt, két darab gép felhasználásával. Az adatbázist és az alkalmazást kiszolgáló gépben négymagos Intel mikroprocesszor és 8 GB memória (RAM) volt. Ezen a gépen Linux Linux operációs rendszer fut (kernel v. 2.6), a http kapcsolatok kiszolgálását az Apache (v. 2.2.9) webszerver biztosítja. Az alkalmazási réteg a PHP (v. 5.2.6) parancsnyelvi környezeten alapul, az adatbázis-kiszolgáló pedig a MySQL (v. 5.1).

A hallgatókat jelképező parancsfájlok egy másik gépen futottak, mert a több ezer példányban elindított parancssori böngésző alapvetően befolyásolta volna az elvégzendő mérés eredményeit, ha azok is az adatbázis-kiszolgálón futottak volna. Ez a gép egy kétmagos processzort és mindösszesen 1 GB memóriát tartalmaz, de ez a választás esetleges volt, egyetlen megkötést lehet tenni: kell tudnia futtatni elegendő számú parancssori böngészőt. Ha ez nem lenne megoldható, ak-

224 L. 148. old.

kor arányosan több gépre kell elosztani azok futtatását, de ez a probléma nem merült föl.

A méréshez felhasznált adatbázisban a mérési eredmények összehasonlíthatósága érdekében olyan mennyiségű tesztadat volt, amely összevethető főiskolánk mennyiségi mutatóival. 8.192 hallgató, átlagosan 4 vizsgakötelezettséggel fejenként. 640 vizsgaalkalom egyenként 80 fős létszámkorláttal. Ez összesen 51.200 vizsgahelyet jelent, míg a minimálisan szükséges vizsgahelyek száma 32.768 lenne. Ez kicsit több mint másfélszeres (1,5625-szörös) túlbiztosítást jelent, ami összhangban van főiskolánk írott és íratlan szabályaival, miszerint a tényleges vizsgalétszám másfélszeresének megfelelő mennyiségű vizsgaalkalmat kell meghirdetni. Esetünkben ennek az a jelentősége, hogy a véletlenszerű időpontválasztás jó eséllyel kevés kísérletből lesz sikeres.

A mérés során használt tesztalkalmazás kifejlesztésének összefoglalóját Szikora Péter kollégám részletesen ismertette a 7. MEB konferencián. [F4367/A] A mérés alapjául szolgáló adatmodell kidolgozásának részleteit és tanulságait ugyanezen konferencián én magam ismertetem. [F4367/B]

Mivel a vizsgajelentkezési időszak megnyílásakor fellépő csúcsterhelés a problémás, ezért a mérés során olyan helyzetre volt szükség, amelyben minél rövidebb idő alatt minél több jelentkezési kísérlet történik. Az esetleges várakozásokra tekintettel a parancssori böngészők beállításai a következők voltak: legföljebb 4 próbálkozás, 30 másodperc a legtöbb várakozási idő egy próbálkozás során, két próbálkozás között 10-30 másodperc közötti véletlenszerű hosszúságú szünet. Ezen beállítás megfelelhet egy emberileg még (éppen) elviselhető esetnek.

A fenti beállítású parancssori böngészők indítása egy parancsfájlból történt, mégpedig úgy, hogy minden 100 böngészőindítás után 2 másodperc szünet következett. Ebből adódik, hogy az összes, 8.192 jelentkezés elindításához szükséges idő annyival több 162 másodpercnél, amennyi időt az operációs rendszer a böngészők futtatásra való felkészítésével és esetleges járulékos tevékenységekkel tölt el. Parncsscori böngészőnek a wget programot használtam.

A parancsfájl az alábbi:

```
#!/bin/bash
#
hova="logV_1c"
((min_i=1))
((max_i=8192))
```

```
mkdir -p ./$hova/wget/

# load naplózás indítása a kiszolgálón:
wget -b -o /dev/null -O /dev/null  »»
  »» http://syxtus.banki.hu/nimrod/start.php? »»
  »» file=${hova}\&sec=360

((i=min_i))
while ((i<=max_i)) ; do
   felvezet=""
   if ((i<1000)) ; then felvezet="0"
   fi
   if ((i<100))  ; then felvezet="00"
   fi
   if ((i<10))   ; then felvezet="000"
   fi

   let j=i/100
   let k=j*100
   if [ $k = $i ] ; then
      sleep 2
   fi

wget -a $hova/wg_p.log -b -t 4 -T 30 -w 20  »»
  »» --random-wait  »»
  »» -P $hova/wget/B$felvezet$i   »»
  »» http://193.225.224.199/nimrod/index.php\?userid=$i

   ((i=i+1))

done
```

A parancsfájl lefutása után a B0001..B8192 alkönyvtárakban tárolt 'index.php?userid=n' (n=1..8192) fájlok tartalma is mutatja a jelentkezés sikerességét illetve sikertelenségét. Sikeres jelentkezés esetén a fájl tartalmazza a 'SZABAD' szövegelemet, majd tárgyanként a tényleges vizsgaidőpontokat. A wget naplójában a sikeres jelentkezéshez az alábbi mintájú bejegyzések tartoznak:

```
2009-05-12 14:34:19--  http://193.225.224.199/nimrod/»»
  »»index.php?userid=7822
2009-02-11 14:34:20 (...) index.php?userid=7822 saved
```

Sikertelen kísérlet esetén a második elem hiányzik a naplóból. A wget naplófájl ez alapján kiértékelhető, tartalma a helyben mentett fájlok tartalmával gépi úton összevethető, ami ellenőrzési lehetőséget jelent.

A mért adatok értékelése, következtetések

A fenti körülmények között elvégzett mérés eredménye messze jobb a vártnál. 3 perc 7 másodperc[225] alatt lezajlott az egész folyamat az adatbázist és az alkalmazást kiszolgáló gép alacsony terhelési szintje mellett. A leghosszabb jelentkezés ideje sem haladta túl a másfél másodpercet, ebből következően nem lehetett sikertelen jelentkezési kísérlet (és nem is volt).

Ez azt jelenti, hogy 8.192 hallgató összesen 32.768 vizsgajelentkezése lezajlott ezen 187 másodperc alatt, egy teszt-hallgató összes vizsgajelentkezésének lebonyolítása pedig átlagosan 0,0228 másodpercet vett igénybe. A leghosszabb naplózott időtartam sem volt 1 (egy) másodpercnél hosszabb. Ez az érték nyilvánvalóan pontatlan, hiszen a wget másodperc pontossággal naplózza az időt, tehát a kerekítési hibára tekintettel a leghosszabb időtartam nem érhette el a tényleges 1,5 másodpercet. A pontos érték ismerete azonban a fentebb már kifejtettek miatt nem különösebben érdekes számunkra.[226]

A mért értékek - a mai viszonyok figyelembe vételével egészen bizonyosan - a 'nagyon jó' kategóriába tartoznak. Nem hagyható figyelmen kívül az a körülmény, hogy a kiszolgáló gép négymagos processzora és 8 GB memóriája PC kategóriában ugyan igen jónak tekinthető, de kiszolgálóként (szerver) egyáltalán nem kiugró, továbbá az a körülmény sem, hogy az operatív memória napjainkban olcsó erőforrás: 4 GB 1.066 MHz DDR2 RAM modul 2009-ben bruttó 16 és 20 ezer forint között mozog.

Fontos körülmény, hogy a mérési környezet egyik elemének sajátos teljesítményfokozó lehetőségeit sem használtam ki a mérés során. Az adatbázis szerkezetében kizárólag az elsődleges kulcsokra - egyébként

225 Ez a legjobb idő. A legrosszabb mért idő 3 perc 9 másodperc volt. Ez 1,07%-os eltérést jelent, ami nem tekinthető számottevőnek.

226 L. a Mit és hogyan érdemes mérni c. részt, 176. old.

önműködően létrehozott - indexek szerepelnek. A PHP programfájl értelmező módban fut, nincs előre lefordítva, holott erre lenne technikai lehetőség (Zend engine), de ez már az általános érvényt érdemben korlátozná. Ennélfogva a teszt-adatbázis és maga a mérési eljárás hordozhatósága a lehető legmagasabb szintű, sajnos a különféle operációs rendszerek, programnyelvek, adatbáziskezelők és webkiszolgálók sajátosságai okán így sem lehet egy az egyben tetszőleges más környezetre átvinni a mérést, azonban még mindig így sikerül ehhez a legközelebb jutni.

Ezen körülmények között a mérési eredmények alátámasztják azt, hogy a megfelelő minőségű háromszintű adatmodellezés és tervezés mindenképpen szükséges (de nem elégséges) előfeltétele a jó eredmények elérésének. Különösen fontos a redundanciamentességre való törekvés. Nem véletlen, hogy Date 2006-ban megjelent összegző munkájában két teljes fejezetet szentel a redundancia kérdésének, Redundáns adatok és adatbázistervezés, További gondolatok címen. [F4281 pp. 217-254., 255-270.]

Hangsúlyoznom kell azonban, hogy ez a mérés bár alátámasztja, de matematikai vagy fizikai értelemben véve nem bizonyítja állításomat. Már csak azért sem, mert a végeredményeket az adatmodell jóságán kívül számos egyéb tényező befolyásolja, amely tényezők hatásai egymást ellensúlyozhatják, és mert a befolyásoló tényezők közül számos - mindjárt maga a fogalmi modell minősége - önmagában, matematikai értelemben nem számszerűsíthető.

A fogalmi szintű adatmodell ugyan elsődleges fontosságú tényező, mint amelyen az egész adatbázis és a köré épülő információs rendszer nyugszik, de nem kizárólagos. Mivel a végeredményt nagyszámú tényező határozza meg - ezek hatásai között akár nagyságrendnyi különbségek is lehetnek -, legalább az elsődleges fontosságú tényezők pontos szerepét meg kellene tudni határozni. Ezek - a fogalmi modell minőségén túl - a logikai szintű tervezésnek és a fizikai szintű tényezőknek a hatékonyságra gyakorolt hatása, a hardver teljesítménye, az adatbáziskezelő kiválasztása, az alkalmazás programkódjának minősége és az adatmodellek különbözősége (a redundanciamentességen túl).

Esetemben azt kellene tudni indokolni, hogy ezen tényezők nem gyakorolhatnak olyan mértékű hatást a mérési eredményekre, amelyek alapjaiban kérdőjelezhetnék meg a végkövetkeztetést, azaz ezen tényezők egyike sem, illetve azok együttesen sem lehetnek olyan hatással a hatékonyságra, amely hatás ellensúlyozhatná a fogalmi modell jóságának alapvetően meghatározó hatását.

Nehezíti helyzetemet, hogy az összehasonlítási alapul szolgáló Neptunt nem volt módom módszeresen tesztelni, annak adatmodelljéhez és adatterveihez nincs és sajnos nem is lehet hozzáférésem, és az alkalmazás forráskódját sem tudom vizsgálni és elemezni. Mindevvel együtt is tehetők megállapítások.

Logikai szintű tervezés: A logikai szintű tervezés során semmilyen olyan megoldást nem terveztem, amely a hatékonyság közvetlen növelését szolgálná, bár ilyen lehetőség van.[227]

Fizikai szintű tervezés: Ennek során az elsődleges kulcsokra vonatkozó - a kezelő által automatikusan létrehozott - indexelésen túl semmilyen lehetőséget nem használtam ki, amely a hatékonyságot közvetlenül növelhetné (partícionálás, szegmentálás, fürtözés).

Hardverteljesítmény: Az általam használt kísérleti környezet hardverteljesítménye lényegesen alacsonyabb, mint a Neptuné.

Adatbáziskezelő: esetemben MySQL, a főiskolai Neptun esetében Oracle. Az adatbáziskezelő kiválasztása során semmilyen tényszerű kényszerítő körülmény nincs sem az én esetemben, sem a Neptun esetében. Az Oracle professzionálisabb kezelőnek számít mind történelmi múltja, mind referenciái alapján, mint a MySQL, de inkább a nagy (sokkal nagyobb) adatbázisok esetében. Érdemes lenne megismételni a méréseimet Oracle adatbáziskezelő alatt a számszerű eredmények összehasonlítása céljából, de ez jelenlegi lehetőségeimet meghaladja.

Az alkalmazás programkódja: érdemben nem befolyásolja az eredményt, ha mégis, akkor az igen komoly programozási hibák együttes jelenlétével magyarázható csak. Az adatbázis teljesítményét az alkalmazás programnyelve - közvetlenül legalábbis - nem befolyásolja. Evvel együtt is lehet szűk keresztmetszet. A Neptun esetében több alkalmazásszerver van az adatbázisszerver „előtt". Ezen túl mindkét adatbáziskezelő a definitív SQL alapján működik, tehát az alkalmazási rétegben adatbáziskezeléssel összefüggő érdemi tevékenység nem történik. Így tehát az alkalmazási réteg szűk keresztmetszet mivolta megkérdőjelezhető, illetve ha mégis az, akkor ez olyan programozási, hardverméretezési vagy koncepcionális problémákat takar, amelyeket az adatbázistól függetlenül meg kell oldani, pontosabban már régen meg kellett volna oldani.

Adatmodellek különbözősége: egyrészt pont ezen van a hangsúly, ennek döntő szerepét kell alátámasztanom gondolatmenetem igazolására. A Neptun adatbázisának mérete alapján és közvetlen SQL-lekér-

227 L. az „A hatékonyság növelésére..." kezdetű bekezdést a Logikai szintű tervezés c. részben, 149. old.

dezéseket futtató kollégák tapasztalatai alapján föltételezhető, hogy annak adatmodellje erősen redundáns. Nyilvánvaló, hogy kisebb adatbázis kezelése (lényegesen) gyorsabb mint egy nagyobb adatbázisé. Mivel a redundancia egyértelműen modellezési hiba (kivéve az eseti, tudatos logikai szintű döntéseket, amikor tárterületet áldozunk sebességnövekedésért cserébe), azt kell biztosítanom, hogy az általam tervezett adatmodell elegendően bonyolult és elegendően nagy adattömegű legyen, még akkor is, ha részleteiben csak a vizsgajelentkezéshez szükséges részt terveztem meg. Ez olymódon történt, hogy minden további lényeges részterületet - annak pontos és részletes megtervezése nélkül - a jellemző mennyiségű táblával és azokban jellemző mennyiségű adattal szerepeltetek az adatbázisban. A biztonság irányába való elmozdulás céljából mind a táblák mennyiségét, mind az egyes táblákban szereplő tesztadat mennyiségét a becsült értékhez képest 150%-os „biztonsági tényezővel” szoroztam.

6. A dolgozat elkészítésének módszerei és eszközei

A dolgozat elkészítése önmagában is számos tanulsággal járt. Ezen tanulságok köre a természetes szakmai körön túl kiterjed az alkalmazni kívánt és alkalmazott módszerek és eszközök sajátosságain túl arra is, hogy az eredeti elgondolásaimat számos esetben módosítanom kellett menet közben. Ennek oka adott esetben az volt, hogy előzetes elgondolásaim, sőt elvárásaim menet közben, a munka előrehaladtával finomodtak, alakultak. Más esetekben a számomra elérhető erőforrások korlátaival kellett számot vetnem. Nem sikerült hozzájutnom több olyan eszközhöz, amelyekhez szerettem volna, és amelyek kipróbálása, illetve alkalmazása a képet árnyaltabbá tette volna.

Ma már a témaválasztás is némi magyarázatot igényel, legalábbis az SSADM-re vonatkozólag. Az SSADM történetének ismertetése során szerepelt, hogy az SSADM „de facto" elhalása az ezredforduló utánra datálható.[228] Amikor a témával elkezdtem foglalkozni, erről még nem volt szó. Fölmerül az a kérdés, hogy nem kellett volna-e ezt, vagy legalábbis ennek lehetőségét előre látnom a folyamatok észlelhető iránya alapján. Véleményem szerint ez nem befolyásolja lényegesen a tárgyaltakat. Mondanivalóm lényege ugyanis nem szorosan az SSADM-hez kötődik, számos más fejlesztési módszert is választhattam volna „állatorvosi lónak", mert a háromszintű modellezés méltatlanul a háttérbe szorul mind a mai napig.

Ugyanakkor az SSADM módszer számos kiváló technika alkalmazásával segíti elő a hatékony munkavégzést, és megítélésem szerint méltatlanul szorult a háttérbe az elmúlt időszakban. Ezt a véleményemet támasztja alá az a két körülmény is, hogy hazánkban mind a mai napig kormányzati ajánlás vonatkozik a használatára,[229] továbbá hogy dokumentáltan használták még a közelmúltban is.[230]

A manapság szakmai standardnak számító, objektumorientált UML jóval későbbi lehetőség. Számos gyermekbetegséggel terhelt 1.1-es változatát 1997-ben publikálták, hivatalos szabvánnyá pedig a közelmúltban vált.[231]

228 L. Az SSADM történetéről szóló részben, 79. old.
229 L. az „Az SSADM-mel Magyarországon..." kezdetű bekezdést, 80. old.
230 L. az „Az SSADM mint rendszerelemzési..." kezdetű bekezdést, 80. old.
231 ISO/IEC 19501:2005 Information technology — Open Distributed Processing — Unified Modeling Language (UML) Version 1.4.2

Megemlítendő még az a körülmény, hogy a manapság szinte kizárólagossá váló objektumorientált szemlélet és az adatbáziskezelés relációs elmélete nem hozható közös nevezőre nehézségek nélkül, következésképp az objektumorientált tervezési és fejlesztési módszer(ek) alkalmazása a korábbiakhoz képest más jellegű, újabb problémákat vet föl, miközben a korábbi problémák közül sem oldja meg mindet.

Ezen körülmények között megmaradtam az eredetileg jóváhagyott témaválasztásnál, és minimális, elkerülhetetlen kivételektől eltekintve az eredetileg jóváhagyott témavázlatnál.

Alkalmazott módszerek és korlátaik

Dolgozatom elkészítése során feltártam és feldolgoztam a mértékadó szakirodalom leglényegesebb részeit a kezdetektől napjainkig az elméleti háttér tisztázásához. Ehhez saját adatbázist és alkalmazást terveztem és fejlesztettem. Felhasználtam a felsőoktatásban eddig (hallgatóként és oktatóként) eltöltött több mint két évtized személyes tapasztalatait és különféle részterületeken gyűjtött szakmai tapasztalataimat egyaránt. Konzultációkat folytattam egyes kérdésekben a mértékadó beosztásban dolgozó kollégákkal, bár ezt a helyzet sajátosságaira tekintettel igyekeztem a lehető legszükségesebbre korlátozni. Áttekintettem a közvetlenül vonatkozó jogszabályokat. Kísérleteket és méréseket terveztem és végeztem mondanivalóm alátámasztására.

Hivatkozások

Szakmailag közismert alapdolgokra általában nem hivatkozom, hacsak az elsődleges mondanivaló ezt nem teszi feltétlenül szükségessé.

Az első részben, az alapfogalmak tisztázása során idézett példáknál szándékosan nem szerepel forrásmegjelölés. Ha a tételes és teljeskörű fogalmi és szóhasználati összehasonlítás lenne a dolgozat tárgya, például az egységes és nyelvileg megfelelő színvonalú szaknyelvi alapfogalom-rendszer kialakítására tett javaslat céljából, akkor ez nyilván nem lenne lehetséges. Minden idézet pontos forrása és azon belüli helye természetesen dokumentált, de ezen példák véletlenszerűen kiragadottak, és pusztán illusztrációként szerepelnek. Nem volna tisztességes eljárás részemről, ha egyébként általam is nagyra becsült szerzőik szakmai hozzáértését akár csak látszólag is kétségbe vonnám. Annál is inkább, mivel közülük többek munkáit felhasználtam a későbbiek folyamán. Ezen túlmenően még azt sem állíthatom, hogy az általam a bevezetésben körülírt alapfogalmak, azok rendszere és szóhasználata ne

lenne esetleg szintén vitatható. Mindenesetre a dolgozat megírása során törekedtem azok következetes, a nekik tulajdonított jelentéstartalomnak pontosan megfelelő használatára.

A szakirodalmi forrásokra történő szövegközi hivatkozásokban szereplő sorszámok a saját forrásadatbázisbeli azonosítók értékei.

Törekedtem arra, hogy dolgozatomban az indokolt helyeken keresztthivatkozásokat helyezzek el a már tárgyalt részekre.

Szakirodalmi források

Mivel dolgozatom tárgya nem történeti jellegű, szövegkritikai szempontok nem merülnek föl, ezért általában nem nélkülözhetetlen az elsődleges források használata. Például Ted Codd, a relációs adatbáziskezelés megalapozójának gyűjteményes kötete (annak is második változata 1990-ből) tökéletesen megfelelt céljaimnak, az eredeti első közlésű tanulmányok felkutatása az 1970-es évekbeli (sőt 1969-beli) amerikai folyóiratokból aránytalanul nehéz lenne anélkül, hogy - esetemben - bármilyen érdemi többlettel szolgálhatna. Ahogy Codd írja az előszóban, könyvében egybegyűjti korábbi publikációinak legtöbbjét, többletként számos új szemponttal és bővebb magyarázatokkal.[232] [F4296 p. vi.])

Az általam felhasznált szakirodalmi forrásokat fő csoportjai:

A szakterületet megalapozó tudósok eredeti elméleti munkái vagy azok szempontomból egyenértékű későbbi, akár bővített kiadásai.

A szakterület irodalmának mai (közelmúltbeli) ismertebb elméleti vonatkozású munkái. Ezek elsődleges tanulsága számomra az, hogy a területet megalapozó tudósok harminc-negyven évvel ezelőtt közel tökéletes és közel teljes munkát végeztek. Sok, illetve érdemi újdonságot, alapvető újítást nem igazán sikerült hozzátenni a relációs adatbáziskezelés elméletéhez, igaz, hogy erre alig van (vagy egyáltalán nincs) szükség. Ez nem azt jelenti, hogy akár fontos részletkérdésekben ne lennének szakmai viták, l. pl. az üres értékek problémakörét, amely a kezdetektől egészen napjainkig nem megoldott.[233]

A szakterülethez kapcsolódó más szakterületek mint esetünkben segédtudományok szakirodalma.

Egyéb, elsősorban technikai jellegű kérdésekre, részletekre, vonatkozó irodalom, beleértve pl. vonatkozó jogszabályokat is.

232 „This book collects in one document much of what has appeared in my technical papers, but with numerous new features, plus more detailed explanation (and some emphasis."

233 L. az Üres értékek c. részt, 48. old.

Eszközök

Különösen is szeretném megemlíteni, hogy komoly, ipari szintű CASE-eszközökhöz nem sikerült hozzájutnom, ennélfogva nem sikerülhetett ezeket érdemi módon összehasonlítani, következésképp nem egy tartalmilag megalapozott összehasonlító értékelés alapján választottam, hanem az alapján, hogy mi volt elérhető egyáltalán számomra. Igen előnyös lett volna (sőt lenne) legalább a 3-4 legelterjedtebb adatbáziskezelő készségszintű, a napi munka gyakorlatából való készségszintű ismerete. A MySQL adatbáziskezelőre mint legkönnyebben hozzáférhető, ugyanakkor széles körben elterjedt eszközre esett a választás.

Magának a dolgozatnak a szövegszerű értelemben vett elkészítéséhez az OpenOffice.org szövegszerkesztőjét használtam. A végzett munka napi mentése kétféleképpen történt. A mindenkori legfrissebb változatot munkahelyi gépemre másoltam. Emellett egy pendrive-ra lépcsőzetes mentést készítettem, azaz itt minden egyes korábbi változat is megtalálható.

Nagyon fontosnak tartom a dolgozat végleges változatának kétoldalas nyomtatását mint a környezetvédelem jogos szempontjainak érvényesítését. Ezen túlmenően érdemes megállapítani, hogy az egyoldalasságot a mechanikus írógépek technikai sajátosságai követelték meg, és innen hagyományozódott a lézernyomtatók korára. A papír mechanikai átütésén túl ugyanis semmi nem indokolja az egyoldalas nyomtatást.

Saját tervezésű cédulaadatbázis

A cédulázás régi eszköze a fontos, szükséges adatok rendszerezett nyilvántartásának abból a célból, hogy azokat később, szükség esetén, könnyen, vagy legalábbis könnyebben elő lehessen venni és felhasználni. Umberto Eco részletesen tárgyalja a bibliográfiák összeállításának kapcsán. [F4380 pp. 80-104.]

A cédulázás elsőrendű eszköze a tudásunk alapjául szolgáló adatoknak a rendezgetéséhez és rendszerezéséhez. Nem elvárható ugyanis, hogy valaki fejből ismerje a teljesség igényével és részleteiben bármely adott (rész)terület adatait, forrásait és hivatkozásait.

A hosszú távon eredményes kutatómunka egyik előfeltétele a kutató számára fontos ismeretek és azok lelőhelyeinek gyűjtése. Egyrészt célirányosan, tervszerű kutatómunka során, másrészt pedig nem szabad

lebecsülni a véletlen szerepét sem. Megtörténhet, hogy az ember valami roppant érdekes adatra bukkan, miközben valami mást keres.

Magam gyerekkoromban találkoztam először a cédulázással Halason, Nagy Szeder Istvánnál, családfakutatás során. Emlékszem, elhűlve néztem A/6 méretű céduláinak szépen dobozolt ezreit, mind-mind valamilyen helytörténeti vonatkozású adattal. Ő tanítgatott a cédulázás alapjaira valamikor tíz-tizenkét éves koromban.

Külön tanulmányt lehetne írni a tartalmi feltárás emberi és gépi eszközökkel történő megvalósítási kísérleteiről, de a mai napig nincs (szerintem nem is lesz) a teljesség igényével és általánosan alkalmazható megoldás. A relevanciát ugyanis maga a kutató állapítja meg, ő dönti el, hogy egy adott adat releváns-e -- számára és az adott kutatás szempontjából. Maga a szó eredeti jelentése („a tárgyhoz tartozó”) is világosan mutatja ezt.

Ráadásul még világhálós napjainkban is olyan mennyiségi növekedést mutat a papíralapú szakirodalom is, hogy annak érdemi feldolgozását egyre kevésbé lehet győzni. „...az OSZK munkatársainak az évente beérkező 6-7.000 féle folyóirat bölcsészettudományi tárgyú cikkeinek feldolgozását 2002-ben -- anyagi okok miatt -- be kellett fejezni.” [F4381 p. 318.] Ez nem kizárólag hazai -- a gazdasági-pénzügyi helyzet közismert romlásával magyarázható -- probléma, de már magának a katalogizálásnak a végét is látni vélik egyes szerzők. [F4382 pp. 347-349.]

A digitális tartalmak feltárásáról ekkor még nem is beszéltünk. 2008 júliusi adat szerint a Google 1 trillió weboldalt indexel, és ez nyilván kevesebb az összesnél. Külön kérdés, hogy a Google vajon amerikai (10^12) vagy európai (10^18) értelemben használja a trillió fogalmát... [F4383]

Egy cédulán sok dolog szerepelhet. Pontos idézet, tartalmi idézet, megállapítás (tétel), saját vélemény vagy gondolat, hivatkozás képre, rajzra, hanganyagra stb., emellett tartalmazhat technikai jellegű utalásokat, mint pl. hogy a hivatkozott ábrát retusálni kell még stb.

Minden cédula legfontosabb eleme egy szabad szöveges leírás, ezt kissé tudománytalan módon *Sódernek* neveztem, továbbá van *Szerző* és *Cím*. Utóbbi kettő nem feltétlenül kap minden cédula esetében értéket, sőt az is előfordulhat, hogy fiktív értéket adok meg (ilyenkor zárójelbe teszem). A feljegyzéshez tartozik egy, ugyancsak szabad szöveges leírás *Megjegyzés* néven, amelyben az adott ismerettel kapcsolatos saját megjegyzéseimet, technikai észrevételeimet, további tennivalói-

mat szerepeltetem, amelyek nem részei a cédulázott ismeretnek, de ahhoz közvetlenül kapcsolódnak.

Nélkülözhetetlen tartalmi eleme a cédulának a rajta szereplő ismeret forrása is, beleértve annak a forrásbeli helyét is.

Mivel számítógépes (korunk egy gyakori kifejezésével élve: virtuális) cédulákról van szó, múlhatatlanul szükséges, hogy a cédulán szereplő ismeretekhez kapcsolódóan egy vagy több fájlra is lehessen hivatkozni.

A későbbi csoportosíthatóság és visszakereshetőség céljából tárgyszavakat lehet hozzárendelni a cédulán szereplő ismerethez, mégpedig akárhányat (de véges sokat). Igen fontos, hogy a tárgyszavak rendszere jól átgondolt legyen. Mivel számítógépes cédulákról van szó, szabad kulcsszavas keresést az összes szöveges tartalmú rovatra a számítógép roppant sebességgel tud végezni.

A tárgyszavak alá-fölérendeltségi viszonyba soroltak, azaz egy vagy több fába rendezettek. Ez megkönnyíti a későbbiek során a szűkebb vagy bővebb témák szerint csoportosítandó listázásokat, vagy az akár teljesen különböző területek elkülönítését is. Emellett könnyen megoldható az ideiglenes vagy eseti kulcsszavak hozzárendelése is.

Forrásaink sokfélék lehetnek, hogy csak néhány fajtát vegyünk számba: lehet könyv, folyóirat, konferencia, világhálós oldal stb., nem beszélve a különlegesebbekről, mint pl. kézirat, oklevél, (régészeti) tárgy, szóbeli közlés, miegymás. Ezt még bonyolítja az, hogy a forrás típusbesorolása változhat a körülmények függvényében. Lehet forrás pl. a TMT mint folyóirat, a TMT adott száma, de lehet forrás az abban megjelent valamely tanulmány is.

A források sokfélesége miatt azok adatainak megadása nyilván nem lehet fajtától függetlenül egységes, hanem pont ellenkezőleg: fajtára jellemző. Ezt kellően rugalmas, felhasználói szinten változtatható, bővíthető módon oldottam meg, úgy, hogy felhasználóként lehet megadni az egyes forrásfajták megnevezéseit és az azokhoz tartozó, fontos tulajdonságok neveit is. Az újabb cédula felvételekor pedig két lehetőségünk van: vagy egy már nyilvántartott forrásra utalunk, vagy pedig új forrás megadása is szükséges. Utóbbi esetben a forrás megadásának első lépése a forrás fajtájának kijelölése, majd az ettől függő további jellemzők megadása.

A forrás fajtájára jellemző tulajdonságok változatossága mellett az adott cédulán szereplő ismeretnek az adott forrásban való helye is fajtától függően írható le. Ezért a forráson belüli (a forrás fajtájától függő) helymeghatározás elnevezéseit (pl. évf./szám) is felhasználóként lehet megadni, forrásfajtánként külön.

A különféle forrásfajták aránylag szabadon bővíthető leírását úgy oldottam meg, hogy felhasználói szinten adhatók meg a forrásfajták nevei, valamint az azokat jellemző (a felhasználó számára fontos) tulajdonságok elnevezései is. Egy adott forrás leírásakor létrejön annyi forrástulajdonság-név és forrástulajdonság-érték adatpár, ahány tulajdonsága fontosnak számít, pontosabban annyi, ahány tulajdonságnak értéket adok az adott esetben.

7. Irodalomjegyzék

A szögletes zárójelben szereplő sorszámok az egyes tételek adatbázisbeli azonosítói, a szövegközi hivatkozásoknál is ez szerepel. Fajtánként és ábécésorrendben l. lentebb.

[F4260] Halassy Béla dr.: Az adatbázistervezés alapjai és titkai
IDG Magyarországi Lapkiadó Kft., Budapest, 1995.

[F4261] Dependency Structures of Data Base Relationships
Proc. IFIP Congress, 1974.

[F4262] Abiteboul S. - Hull R. - Vianu Victor: Foundations of Databases
Addison-Wesley Longman, Boston, 1995.

[F4263] Ullman J. - Widom J.: Adatbázisrendszerek - Alapvetés
Panem Könyvkiadó, Budapest, 1998.

[F4265] Falkenberg E.: Significations: The Key to Unify Data Base Management. In: Information Systems, 1976., II. évf. 1. szám

[F4266] Quittner P.: Adatbázis-kezelés a gyakorlatban
Akadémiai Kiadó, Budapest, 1993.

[F4269] Griethuysen (szerk.): Concepts and Terminology for the Conceptual Schema and Information Base
ANSI, h.n., 1982.

[F4271] Codd, E. F.: A Relational Model of Data for Large Shared Data Banks. In: CACM, 1970., XIII. évf. 6. szám

[F4272] Date, C. J.: An Introduction to Database Systems
Addison-Wesley, Reading, 1981.

[F4276] Halassy Béla dr.: Adatmodellezésen alapuló kódtervezés
Számalk, Budapest, 1984.

[F4277] Halassy Béla dr.: Adatbázisok kezelésének alapvető kérdései
Számok, Budapest, 1978.

[F4278] Halassy Béla dr.: Adatmodellezés a rendszerfejlesztésben
Számalk, Budapest, 1983.

[F4279] Halassy Béla dr.: Adatmodellezés, adatbázistervezés
Számok, Budapest, 1980.

[F4280] Date, C. J.: Database in Depth: Relational Theory for Practitioners
O'Reilly, h.n., 2005.

[F4281] Date, C. J.: Date on Database -- Writings 2000-2006
Apress, h.n., 2006.

[F4282] Raffai Mária dr.: Információrendszerek fejlesztése és menedzselése
Novadat Bt., Győr, 2003.

[F4285] Raffai Mária dr.: Az információ - Szerep, hatás, információmenedzsment
Palatia Kiadó, Győr, 2007.
[F4287] Raffai Mária dr.: Adatbázis-tervezés. Modellezés - Fizikai szint
Novadat, Győr, 2006.
[F4288] Kristóf Juli: Barátom, Neptun
http://oktatas.origo.hu/20090204/baratom_a_neptun
letöltés: 2009.02.06.
[F4289] Szücs Ervin: Technika és rendszer
Tankönyvkiadó, Budapest, 1981.
[F4290] Szücs Ervin: Hasonlóság és modell
Műszaki Könyvkiadó, Budapest, 1972.
[F4291] Szücs Ervin: Similitude and Modeling
Elsevier, Amsterdam, 1980.
[F4292] Halassy Béla: Adatmodellezés
Nemzeti Tankönyvkiadó Rt., Budapest, 2002.
[F4296] Codd E. F.: The Relational Model for Database Management - version 2.
Addison-Wesley Publishing Company, h.n., 1990.
[F4297] Sebestyén György dr.: Légy az információs társadalom polgára!
Eötvös Kiadó, Budapest, 2002.
[F4298] Chen P.: The entity-relatonship model - A basis for the enterprise view of data. In: National Computer Conference, AFIPS Press, 1977.
[F4299] Chen P.: The Entity-Relationship Model -- Toward a Unified View of Data. In: ACM Transactions on Database Systems (TODS), 1976. március, I. évf. 1. szám
[F4300] Kovácsné Cohner Judit -- Takács Tibor: Ismerkedés az SSADM-mel
Computerbooks, Budapest, 1999.
[F4302] http://www.mysql.com/why-mysql/marketshare/
letöltés: 2008.01.18.
[F4304] http://list.dev.hu/cgi-bin/mailman/listinfo/sq-l; MM
letöltés: 2009.02.03.
[F4305] http://list.dev.hu/cgi-bin/mailman/listinfo/sq-l; Mr Zed
letöltés: 2009.02.03.
[F4306] http://list.dev.hu/cgi-bin/mailman/listinfo/sq-l; Mr Zed
letöltés: 2009.02.03.
[F4312] Grant J.: Null Values in SQL. In: ACM SIGMOD Record, Association for Computing Machinery Special Interest Group on Management of Data, 2008., XXXVII. évf. 9. szám pp. 23-25.

[F4313] (MTA): Magyar Értelmező Kéziszótár
Akadémiai Kiadó, Budapest, 1982.
[F4314] Halassy Béla dr.: Ember - információ - rendszer
IDG Magyarországi Lapkiadó Vállalat, Budapest, 1996.
[F4315] Bakos Ferenc (szerk.): Idegen szavak és kifejezések szótára
Akadémiai Kiadó, Budapest, 1989.
[F4316] Chen P.: The Entity-Relationship Model - Toward a Unified View of Data. In: ACM Transactions on Database Systems, 1976, I. évf. 1. szám
[F4317] 1992. évi LXIII. tv. a személyes adatok védelméről...
http://www.jogiforum.hu/torvenytar/115
letöltés: 2009.02.17.
[F4318] Pickett, Joseph P. et al.: The American Heritage Dictionary of the English Language
Houghton Mifflin Company, Boston, 2000.
online: http://www.bartleby.com/61/51/D0035100.html
[F4319] West Matthew - Fowler Julian: Developing High Quality Data Models
Shell International Limited, London, 2003.
[F4320] Hetényi Pálné (szerk.): Számítástechnika középfokon
OMIKK, Budapest, 1987.
[F4321] Date C. J.: An Introduction to Database Systems
Addison-Wesley, h.n., 2004.
[F4323] Britannica Hungarica Online
J. I. T. Lexikon Kiadó és Terjesztő Kft., Budapest, 2007.
online: http://www.vilagtudasa.hu/base.aspx?azonosito=554
[F4324] Piattini M. G. - Calero C. - Generol M.: Information and Database Quality
Kluwer Academic Publishers, Norwell, 2002.
[F4325] Bachman C. W.: Data Structure Diagrams. In: Data Base (ACM SIGBDP), 1969. február, I. évf. 2. szám
[F4326] Rejtő Jenő: Az elátkozott part
Magvető Könyvkiadó, Budapest, 1964.
[F4327] Codd E. F.: Extending the Database Relational Model to Capture More Meaning. In: ACM Transactions on Database Systems, 1979., IV. évf. 4. szám
[F4328] http://www-03.ibm.com/ibm/history/exhibits/storage/ »» »» storage_3340.html
letöltés: 2009.02.28.

[F4330] Gulyás Sándor: 50 év az adattárolás történetében, avagy miért winchester a winchester? In: Kék Rózsa - az IBM Magyarország ügyfélmagazinja, 2006/4.
[F4331] http://www-03.ibm.com/ibm/history/history/year_1981.html letöltés: 2009.02.28.
[F4332] http://www-03.ibm.com/ibm/history/exhibits/storage/ »» »» storage_3380.html letöltés: 2009.02.28.
[F4333] Smith L. G. - Williams C. U.: PASA(SM): a method for the performance assessment of software architectures. In: Proceedings of the 3rd international workshop on Software and performance, Róma, 2002.
[F4334] Raffai Mária dr.: Információrendszer-tervezés - Modellezés Logikai szint Novadat, Győr, 2000.
[F4335] Raffai Mária dr.: Információrendszer-tervezés - Modellezés Fizikai szint Novadat, Győr, 2003.
[F4336] Raffai Mária dr.: Az információ - Szerep, hatás, menedzsment Palatia Nyomda és Kiadó, Győr, 2006.
[F4337] Kincses László: SSADM strukturált rendszerelmezési és tervezési módszer MTA Információtechnológiai Alapítvány, h.n., 1993.
[F4338] Szabó Árpád: A görög matematika kibontakozása Magvető, Budapest, 1978.
[F4339] Middleton P.: Management of software engineering. In: International Journal of Computer Applications in Technology, 1999., XII. évf. 2-5. szám
[F4340] Tenner A. R. - DeToro I. J.: BPR - Válalati folyamatok újraformálása Műszaki Könyvkiadó, Budapest, 1998.
[F4341] Falus Iván - Ollé János: Az empirikus kutatások gyakorlata. Adatfeldolgozás és statisztikai elemzés. Nemzeti Tankönyvkiadó, Budapest, 2008.
[F4342] Molnár Bálint: Egy átfogó strukturált rendszerelemzési módszertan Budapesti Közgazdaságtudományi Egyetem, Budapest, 1996.
[F4343] http://www.ogc.gov.uk/guidance_itil.asp letöltés: 2009.03.06.
[F4344] http://www.unl.csi.cuny.edu/faqs/software-enginering/tools.html letöltés: 2009.03.06.

[F4346] Kenneth R. - Steinberg D.: New Frontiers in the Design of Experiments. In: Quality Progress, 2006. augusztus
[F4347] Szikora Péter Gábor: Alternatív Tanulmányi Rendszer (ATR) - Vizsgáztató modul tervezése, elkészítése
ELTE Szakdolgozat, Budapest, 2008.
[F4356] Kung H. J. - Case T.: Traditional and Alternative Database Normalization Techniques: Their impact on IS/IT Students. In: International Journal of Information Technology Education, 2004, I. évf. 1. szám
[F4361] Ullman J. D.: Principles of Database Systems
Pitman, London, 1982.
[F4362] Békéssy András -- Demetrovics János: Adatbázis-szerkezetek
Akadémiai Kiadó, Budapest, 2005.
[F4363] Risztics Péter Károly dr.: Éves jelentés 2006.
BME Információtechnológiai Innovációs és Tudásközpont, Budapest, 2006.
[F4364] Smith L. G. -- Williams C. U.: Performance Solutions -- A Practical Guide to Creating Responsive Scalabe Software
Addison-Wesley, Pearson Education, Inc., 2002.
[F4366] Dijkstra E. W.: The Humble Programmer. In: Communications of the ACM, 1972., XV. évf. 10. szám
[F4367/A] Szikora Péter: Measured Performance of an Information System. 7th International Conference on Management, Enterprise and Benchmarking, Budapest, 2009.
[F4367/B] Keszthelyi András: How to Measure an Information System's Efficiency? 7th International Conference on Management, Enterprise and Benchmarking, Budapest, 2009.
[F4369] Keszthelyi András: Information Management in the Higher Education -- the Role and Importance of the Different Technologies. 3rd International Conference on Management, Enterprise and Benchmarking, Budapest, 2005.
[F4371] Dobay Péter: Vállalati információmenedzsment
Nemzeti Tankönyvkiadó Rt., Budapest, 1997.
[F4374] Schlossnagle George: PHP fejlesztés felsőfokon
Kiskapu Kft., Budapest, 2004.
[F4375] Maczó Kálmán dr. - Horváth Elekné dr. (szerk.): Controlling a gyakorlatban
Verlag Dashöfer Szakkiadó Kft., 2001.
[F4376] Kindler József - Papp Ottó: Komplex rendszerek vizsgálata, Budapest, 1977.

[F4377] Flickenger Rob: Linux bevetés közben
Kiskapu Kiadó, Budapest, 2003.
[F4378] Tóth Tibor (szerk.): Minőségmenedzsment és Informatika
Műszaki Könyvkiadó, Budapest, 1999.
[F4380] Eco U.: Hogyan írjunk szakdolgozatot?
Kairosz, Győr, 2000.
[F4381] Kőrös Kata - Somogyi Tamás - Ternai Zita: Adattármustra. Cikkadatbázisok. In: Tudományos és Műszaki Tájékoztatás, 2008/7.
[F4382] A katalogizálás végnapjai közelednek? Ismertetés. Denskin, Alan: "Tomorrow Never Knows": the end of cataloguing? = IFLA Journal, 33. kötet, 3. szám, 2007. pp. 205-209. In: TMT 2008/7. p. 347-349.
[F4383] We knew the web was big... The Official Google Blog
http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html
letöltés: 2008.09.23.
[F4385] Keszthelyi András: Price, Value and Security -- How to Manage a Database on sy's Own. FIKUSZ (Fiatal Kutatók Szimpóziuma), Budapest, 2009.

Fajtánként és ábécésorrendben

Könyvek

[F4313] (MTA): Magyar Értelmező Kéziszótár
Akadémiai Kiadó, Budapest, 1982.
[F4262] Abiteboul S. - Hull R. - Vianu Victor: Foundations of Databases
Addison-Wesley Longman, Boston, 1995.
[F4315] Bakos Ferenc (szerk.): Idegen szavak és kifejezések szótára
Akadémiai Kiadó, Budapest, 1989.
[F4362] Békéssy András -- Demetrovics János: Adatbázis-szerkezetek
Akadémiai Kiadó, Budapest, 2005.
[F4323] Britannica Hungarica Online
J. I. T. Lexikon Kiadó és Terjesztő Kft., Budapest, 2007.
online: http://www.vilagtudasa.hu/base.aspx?azonosito=554
[F4296] Codd E. F.: The Relational Model for Database Management - version 2
Addison-Wesley Publishing Company, h.n., 1990.
[F4321] Date C. J.: An Introduction to Database Systems
Addison-Wesley, h.n., 2004.

[F4272] Date, C. J.: An Introduction to Database Systems
Addison-Wesley, Reading, 1981.
[F4280] Date, C. J.: Database in Depth: Relational Theory for Practitioners
O'Reilly, h.n., 2005.
[F4281] Date, C. J.: Date on Database -- Writings 2000-2006
Apress, h.n., 2006.
[F4261] Dependency Structures of Data Base Relationships
Proc. IFIP Congress, 1974.
[F4371] Dobay Péter: Vállalati információmenedzsment
Nemzeti Tankönyvkiadó Rt., Budapest, 1997.
[F4380] Eco U.: Hogyan írjunk szakdolgozatot?
Kairosz, Győr, 2000.
[F4341] Falus Iván - Ollé János: Az empirikus kutatások gyakorlata. Adatfeldolgozás és statisztikai elemzés.
Nemzeti Tankönyvkiadó, Budapest, 2008.
[F4377] Flickenger Rob: Linux bevetés közben
Kiskapu Kiadó, Budapest, 2003.
[F4269] Griethuysen (szerk.): Concepts and Terminology for the Conceptual Schema and Information Base
ANSI, h.n., 1982.
[F4260] Halassy B.: Az adatbázistervezés alapjai és titkai
IDG Magyarországi Lapkiadó Kft., Budapest, 1995.
[F4277] Halassy Béla dr.: Adatbázisok kezelésének alapvető kérdései
Számok, Budapest, 1978.
[F4278] Halassy Béla dr.: Adatmodellezés a rendszerfejlesztésben
Számalk, Budapest, 1983.
[F4279] Halassy Béla dr.: Adatmodellezés, adatbázistervezés
Számok, Budapest, 1980.
[F4276] Halassy Béla dr.: Adatmodellezésen alapuló kódtervezés
Számalk, Budapest, 1984.
[F4314] Halassy Béla dr.: Ember - információ - rendszer
IDG Magyarországi Lapkiadó Vállalat, Budapest, 1996.
[F4292] Halassy Béla: Adatmodellezés
Nemzeti Tankönyvkiadó Rt., Budapest, 2002.
[F4320] Hetényi Pálné (szerk.): Számítástechnika középfokon
OMIKK, Budapest, 1987.
[F4337] Kincses László: SSADM strukturált rendszerelmezési és tervezési módszer
MTA Információtechnológiai Alapítvány, h.n., 1993.

[F4376] Kindler József - Papp Ottó: Komplex rendszerek vizsgálata, Budapest, 1977.
[F4300] Kovácsné Cohner Judit -- Takács Tibor: Ismerkedés az SSADM-mel Computerbooks, Budapest, 1999.
[F4375] Maczó Kálmán dr. - Horváth Elekné dr. (szerk.): Controlling a gyakorlatban
Verlag Dashöfer Szakkiadó Kft., 2001.
[F4342] Molnár Bálint: Egy átfogó strukturált rendszerelemzési módszertan
Budapesti Közgazdaságtudományi Egyetem, Budapest, 1996.
[F4324] Piattini M. G. - Calero C. - Generol M.: Information and Database Quality
Kluwer Academic Publishers, Norwell, 2002.
[F4318] Pickett, Joseph P. et al.: The American Heritage Dictionary of the English Language
Houghton Mifflin Company, Boston, 2000.
online: http://www.bartleby.com/61/51/D0035100.html
[F4266] Quittner P.: Adatbázis-kezelés a gyakorlatban
Akadémiai Kiadó, Budapest, 1993.
[F4287] Raffai Mária dr.: Adatbázis-tervezés. Modellezés - Fizikai szint
Novadat, Győr, 2006.
[F4285] Raffai Mária dr.: Az információ - Szerep, hatás, információmenedzsment
Palatia Kiadó, Győr, 2007.
[F4336] Raffai Mária dr.: Az információ - Szerep, hatás, menedzsment
Palatia Nyomda és Kiadó, Győr, 2006.
[F4335] Raffai Mária dr.: Információrendszer-tervezés - Modellezés Fizikai szint
Novadat, Győr, 2003.
[F4334] Raffai Mária dr.: Információrendszer-tervezés - Modellezés Logikai szint
Novadat, Győr, 2000.
[F4282] Raffai Mária dr.: Információrendszerek fejlesztése és menedzselése
Novadat Bt., Győr, 2003.
[F4326] Rejtő Jenő: Az elátkozott part
Magvető Könyvkiadó, Budapest, 1964.
[F4363] Risztics Péter Károly dr.: Éves jelentés 2006.
BME Információtechnológiai Innovációs és Tudásközpont, Budapest, 2006.
[F4374] Schlossnagle George: PHP fejlesztés felsőfokon
Kiskapu Kft., Budapest, 2004.

[F4297] Sebestyén György dr.: Légy az információs társadalom polgára!
Eötvös Kiadó, Budapest, 2002.
[F4364] Smith L. G. -- Williams C. U.: Performance Solutions -- A Practical Guide to Creating Responsive Scalabe Software
Addison-Wesley, Pearson Education, Inc., 2002.
[F4338] Szabó Árpád: A görög matematika kibontakozása
Magvető, Budapest, 1978.
[F4347] Szikora Péter Gábor: Alternatív Tanulmányi Rendszer (ATR) - Vizsgáztató modul tervezése, elkészítése
ELTE Szakdolgozat, Budapest, 2008.
[F4290] Szücs Ervin: Hasonlóság és modell
Műszaki Könyvkiadó, Budapest, 1972.
[F4291] Szücs Ervin: Similitude and Modeling
Elsevier, Amsterdam, 1980.
[F4289] Szücs Ervin: Technika és rendszer
Tankönyvkiadó, Budapest, 1981.
[F4340] Tenner A. R. - DeToro I. J.: BPR - Vállalati folyamatok újraformálása
Műszaki Könyvkiadó, Budapest, 1998.
[F4378] Tóth Tibor (szerk.): Minőségmenedzsment és Informatika
Műszaki Könyvkiadó, Budapest, 1999.
[F4263] Ullman J. - Widom J.: Adatbázisrendszerek - Alapvetés
Panem Könyvkiadó, Budapest, 1998.
[F4361] Ullman J. D.: Principles of Database Systems
Pitman, London, 1982.
[F4319] West Matthew - Fowler Julian: Developing High Quality Data Models
Shell International Limited, London, 2003.

Folyóiratok, konferenciák

[F4382] A katalogizálás végnapjai közelednek? Ismertetés. Denskin, Alan: "Tomorrow Never Knows": the end of cataloguing? = IFLA Journal, 33. kötet, 3. szám, 2007. pp. 205-209. In: TMT 2008/7. p. 347-349.
[F4325] Bachman C. W.: Data Structure Diagrams. In: Data Base (ACM SIGBDP), 1969. február, I. évf. 2. szám
[F4299] Chen P.: The Entity-Relationship Model -- Toward a Unified View of Data. In: ACM Transactions on Database Systems (TODS), 1976. március, I. évf. 1. szám
[F4316] Chen P.: The Entity-Relationship Model - Toward a Unified View of Data
ACM Transactions on Database Systems, I. évf. 1. szám, 1976.

[F4298] Chen P.: The entity-relatonship model - A basis for the enterprise view of data. In: National Computer Conference, AFIPS Press, 1977.
[F4327] Codd E. F.: Extending the Database Relational Model to Capture More Meaning. In: ACM Transactions on Database Systems, 1979., IV. évf. 4. szám
[F4271] Codd E. F.: A Relational Model of Data for Large Shared Data Banks. CACM, 1970., XIII. évf. 6. szám
[F4366] Dijkstra E. W.: The Humble Programmer. In: Communications of the ACM, 1972., XV. évf. 10. szám
[F4265] Falkenberg E.: Significations: The Key to Unify Data Base Management. In: Information Systems, 1976., II. évf. 1. szám
[F4312] Grant J.: Null Values in SQL. In: ACM SIGMOD Record, Association for Computing Machinery Special Interest Group on Management of Data, 2008. szeptember, XXXVII. évf. 9. szám
[F4330] Gulyás Sándor: 50 év az adattárolás történetében, avagy miért winchester a winchester? In: Kék Rózsa - az IBM Magyarország ügyfélmagazinja, 2006/4.
[F4346] Kenneth R. - Steinberg D.: New Frontiers in the Design of Experiments. In: Quality Progress, 2006. augusztus
[F4367/B] Keszthelyi András: How to Measure an Information System's Efficiency? 7th International Conference on Management, Enterprise and Benchmarking, Budapest, 2009.
[F4369] Keszthelyi András: Information Management in the Higher Education -- the Role and Importance of the Different Technologies. 3rd International Conference on Management, Enterprise and Benchmarking, Budapest, 2005.
[F4385] Keszthelyi András: Price, Value and Security -- How to Manage a Database on sy's Own. FIKUSZ (Fiatal Kutatók Szimpóziuma), Budapest, 2009.
[F4381] Kőrös Kata - Somogyi Tamás - Ternai Zita: Adattármustra. Cikkadatbázisok. In: Tudományos és Műszaki Tájékoztatás, 2008/7.
[F4356] Kung H. J. - Case T.: Traditional and Alternative Database Normalization Techniques: Their impact on IS/IT Students. In: International Journal of Information Technology Education, 2004/1., I. évf. 1. szám
[F4339] Middleton P.: Management of software engineering. In: International Journal of Computer Applications in Technology, 1999., XII. évf. 2-5. szám

[F4333] Smith L. G. - Williams C. U.: PASA(SM): a method for the performance assessment of software architectures. In: Proceedings of the 3rd international workshop on Software and performance, Róma, 2002.

[F4367/A] Szikora Péter: Measured Performance of an Information System. 7th International Conference on Management, Enterprise and Benchmarking, Budapest, 2009.

Digitális, világhálós források

[F4304] http://list.dev.hu/cgi-bin/mailman/listinfo/sq-l; MM
letöltés: 2009.02.03.

[F4305] http://list.dev.hu/cgi-bin/mailman/listinfo/sq-l; Mr Zed
letöltés: 2009.02.03.

[F4306] http://list.dev.hu/cgi-bin/mailman/listinfo/sq-l; Mr Zed
letöltés: 2009.02.03.

[F4288] http://oktatas.origo.hu/20090204/baratom_a_neptun
letöltés: 2009.02.06.

[F4328] http://www-03.ibm.com/ibm/history/exhibits/storage/ »» »» storage_3340.html
letöltés: 2009.02.28.

[F4332] http://www-03.ibm.com/ibm/history/exhibits/storage/ »» »» storage_3380.html
letöltés: 2009.02.28.

[F4331] http://www-03.ibm.com/ibm/history/history/year_1981.html
letöltés: 2009.02.28.

[F4317] 1992. évi LXIII. tv. a személyes adatok védelméről...
http://www.jogiforum.hu/torvenytar/115
letöltés: 2009.02.17.

[F4302] http://www.mysql.com/why-mysql/marketshare/
letöltés: 2008.01.18.

[F4343] http://www.ogc.gov.uk/guidance_itil.asp
letöltés: 2009.03.06.

[F4344] http://www.unl.csi.cuny.edu/faqs/software-enginering/tools.html
letöltés: 2009.03.06.

[F4383] We knew the web was big... The Official Google Blog
http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html
letöltés: 2008.09.23.